FINAL EVALUATION REPORT

The IBM Corporation

RS/6000 Distributed System

Running

AIX Version 4.3.1 TCSEC Evaluated C2 Security

Prepared by:

Arca Systems

TTAP Evaluation Facility

Foreword

This publication, the *Final Evaluation Report: The IBM Corporation RS/6000 Distributed System running AIX Version 4.3.1 TCSEC Evaluated C2 Security*, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, *Computer Security Evaluation Center*. The purpose of this report is to document the results of the formal evaluation of IBM Corporation's RS/6000 Distributed System running AIX Version 4.3.1 TCSEC Evaluated C2 Security. The requirements stated in this report are taken from the *Department of Defense Trusted Computer System Evaluation Criteria*, dated December 1985.

Acknowledgments

Evaluation Team Members

Douglas J. Landoll
Arca Systems

Diann A. Carpenter
Arca Systems

Christopher J. Romeo
Arca Systems

Suzanne S. McMillion
Computer Security Specialists Inc.

Trademarks

Table of Contents

Executive Summary

The RS/6000 Distributed System running AIX Version 4.3.1 TCSEC Evaluated C2 Security has been evaluated by a team from the Trust Technology Assessment Program (TTAP) Evaluation Facility at Arca Systems. In order to achieve a C2 level of trust rating, the security features of the RS/6000 Distributed System were examined against the requirements specified by the National Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), dated 26 December 1985.

The Arca evaluation team has determined that, when configured as described in the Trusted Facility Manual, the RS/6000 Distributed System satisfies all the specified requirements of the TCSEC at the C2 level of trust.

A C2 level of trust system provides a Trusted Computing Base (TCB) that implements the following:

- User identification and authentication to control general system access;
- Discretionary access control to protect objects and allow users to distribute access to those objects as appropriate; and
- Auditing to provide general user accountability.

User identification and authentication is performed on both interactive login and batch processing. Users are identified by their UID and authenticated by their password. Once a user has successfully logged in to one host in the RS/6000 Distributed System, the user may access other hosts in the distributed system.

Discretionary Access Control (DAC) is provided by a mechanism that allows users to specify and control access to objects that they own. DAC attributes are assigned to an object when it is created and remain in effect until the object is destroyed or its attributes are changed. The DAC policy on each named object can be specified and controlled through permission bits or**,** optionally, through access control lists (ACLs). Permission bits are the standard UNIX DAC mechanism and are used on all RS/6000 Distributed System file system objects. Individual bits are used to indicate permission for read, write, and execute access for the three categories of object users: owner, group and world. There is an optional ACL mechanism that provides a finer level of granularity than permission bits. ACLs can establish separate DAC settings for individual users and groups.

An administrator may specify the auditing of individual events or users. Audit log files are generated based on the audit events selected and the security relevant actions of the users and are generated on each host computer of the RS/6000 Distributed System.

The system architecture of the RS/6000 Distributed System provides a resistant, isolated domain that helps to protect it from external interference or tampering. System integrity tests are available to periodically verify the correct operation of both the hardware and firmware elements of the TCB. The IBM Corporation performed security testing of the security mechanisms against the system design description. The evaluation team reviewed the IBM Corporation testing and

conducted a subset of those tests in addition to its own security tests to ensure that no obvious security flaws exist in the evaluated system.

The Target of Evaluation is a system of IBM RS/6000 host computers connected via a physically protected Local Area Network, communicating via TCP/IP networking. The RS/6000 is a line of high performance Uni-Processor and Symmetric Multi-Processing computers based on 32 bit and 64 bit PowerPC processors. All hosts run the AIX Version 4.3.1 TCSEC Evaluated C2 Security operating system. The Advanced Interactive eXecutive (AIX) operating system is a general-purpose time-sharing operating system based on the AT&T System V UNIX system and incorporates many of the network functions and other enhancements from Berkeley Software Distribution UNIX. AIX is differentiated from other UNIX products by its system administration tools, Journaled File System, pageable/preemptable kernel, loadable kernel extensions, hardware error detection, and available applications.

In summary, the RS/6000 Distributed System running AIX Version 4.3.1 TCSEC Evaluated C2 Security satisfies all requirements at the TCSEC C2 level of trust (C2).

Disclaimer

This document describes features which are unique to the evaluated configuration and which may not be present in other AIX releases or configurations. The description of internal system interfaces, data structures or features does not constitute an agreement by the IBM Corporation to provide those interfaces, data structures or features in future releases of AIX.

1. INTRODUCTION

In October of 1997, during the National Information Systems Security Conference, the National Security Agency (NSA) announced that it would no longer conduct new evaluations of C2 or B1 products and that future product evaluations would be conducted by authorized commercial facilities under the Trust Technology Assessment Program (TTAP). Following the announcement, the NSA signed Cooperative Research and Development Agreements with several commercial laboratories to become authorized commercial facilities under TTAP and allow them to begin C2 and B1 evaluations.

On February 23, 1998, Arca Systems, an authorized TTAP Facility, accepted IBM's RS/6000 Distributed System running AIX Version 4.3.1 TCSEC Evaluated C2 Security for evaluation against the requirements of the TCSEC C2 level of trust.

1.1 Evaluation Process Overview

The Department of Defense Computer Security Center was established within the NSA in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985, the name of the organization was changed to the National Computer Security Center (NCSC). The Trusted Computer System Evaluation Criteria (TCSEC) was written in order to assist in assessing the degree of trust one places in a given computer system. The TCSEC states the specific requirements a computer system must meet in order to achieve a particular level of trustworthiness. The TCSEC levels are arranged hierarchically into four major divisions of protection, each with definite security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class requirements met by a system, the system must be evaluated against the TCSEC by a NSA-approved trusted product evaluation team.

Prior to January 1997, the NCSC conducted all U.S. trusted product evaluations through the Trusted Product Evaluation Program (TPEP). With the advent of TTAP in January 1997, authorized commercial facilities were granted the ability to perform a portion of these evaluations on behalf of the National Security Agency (NSA).

The National Security Agency implemented TTAP in January of 1997 to authorize and oversee commercial facilities performing trusted product evaluations. The principle mechanism for TTAP oversight is the TTAP Oversight Board, which monitors authorized facilities to ensure quality and consistency across evaluations.

Evaluation is an analysis of the hardware and software components of the product, which includes system training, security analysis, and review of the design, user, and test documents. The evaluation team uses the documentation to produce an Initial Product Assessment Report (IPAR) that is reviewed by a Technical Review Board (TRB). The evaluation team briefs the TRB on the product's security architecture and its plan to perform security testing on the product. After the evaluation team performs security testing on the product, the results of security testing are included in a revised IPAR called the Final Evaluation Report (FER). The evaluation team

presents the results of testing at the Final TRB meeting. The TRB then makes recommendations about the product's security entry on the Evaluated Products List (EPL).

1.2 Document Organization

This document is organized as follows:

- Chapter 1 - Introduction
- Chapter 2 - System Overview - high level overview of system security policy and mechanisms.
- Chapter 3 - TCB Interfaces - description of the different types of the TCB interfaces.
- Chapter 4 - TCB Hardware - system hardware architecture.
- Chapter 5 - TCB Software - system software architecture.
- Chapter 6 - TCB Resources - security relevant resources and attributes.
- Chapter 7 - TCB Policies - security policies and mechanisms.
- Chapter 8 - Assurances - additional assurances (e.g., documentation and testing).
- Chapter 9 - Evaluation as a C2 compliant system - evaluation of the product against the applicable TCSEC requirements.
- Appendix A - Evaluated Hardware Components - identification of evaluated hardware
- Appendix B - Evaluated Software - identification of evaluated software.
- Appendix C - Acronyms
- Appendix D - Glossary
- Appendix E - References
- Appendix F - Product Bulletin - EPL entry for the RS/6000 Distributed System.
- Appendix G - Audit Record Format - audit event types.
- Appendix H - Trusted Programs - non-kernel TCB programs.
- Appendix I - Evaluator Comments

1.3 Conventions

The following typographical conventions have been used.

Convention/Use	Description
COMMANDS	Names of commands, processes, and programs (non-kernel) and libraries
pathnames	Names and pathnames of files and directories
System calls	Names of system calls (these invoke the kernel TCB)
Protocols and daemons	Names of network protocols and daemons

1.4 History

Advanced Interactive eXecutive (AIX) is IBM's version of the UNIX operating system. AIX is based on AT&T System V Release 2 and incorporates Berkeley Software Distribution (BSD) enhancements and additional features developed by IBM. The first implementation of AIX goes back over ten years, when AIX was developed for the IBM RT/PC.

The RS/6000 hardware is based on the PowerPC RISC processor, which was produced through a cooperative effort between IBM, Apple, and Motorola. Ancestors of the PowerPC-based RS/6000 include the POWER 9-chip processor module, the POWER2 (a faster, floating-point version of POWER), the RSC (a single-chip version of POWER), and the P2SC (a single-chip version of the POWER2). The first PowerPC-based RS/6000 systems were based on the 601 (the original PowerPC processor) and the 603. AIX still runs on all of these processors, but these earlier versions of the RS/6000 are not included in the ToE.

Table 1-1 summarizes the history of the AIX operating system and the RS/6000 hardware. As shown in the table, external ancestors of AIX include System V, BSD 4.2, and the OSF/1 operating system. The two primary influences on the ToE were AIX Version 3 and the OSF/1 operating system.

Table 1-1. Product History. *The evaluated product evolved from AIX and was heavily influenced by OSF/1 and the TCSEC C2 criteria.*

Date	Event
1984	• AT&T System V.2.
1986	• AIX Version 2 This was the first implementation of AIX, which ran on the RT PC computer. It was derived from the AT&T distribution of System V Release 2 with enhancements from BSD 4.2. AIX ran on a Virtual Resource Manager (VRM) that implemented a virtual machine on top of the RT PC hardware.
1990	• AIX Version 3 This was the first version of AIX to run on the RS/ 6000, using the POWER processor. It maintained the System V Release 2 interface, with BSD 4.3 extensions such as enhanced signals and group lists, and IBM extensions such as mapped files. Capabilities derived from the RT VRM include a kernel that can have portions paged out and can be preempted. New features included the Journaled File System (JFS), support for logical disk partitioning (Logical Volume Manager), dynamically installable kernel extensions, and dynamic program management. • OSF/1 The most recent external influence on AIX was the Open Software Foundation (OSF) and the OSF/1 operating system. As a founding member of OSF, IBM contributed source code (commands, libraries, and Logical Volume Manager) and concepts to OSF/1. In return, IBM used OSF/1 code as the basis for its standardization of AIX Version 4.
1991	• IBM, Apple, Motorola PowerPC alliance formed.
1993	• IBM PowerPC hardware introduced. • IBM Scalable POWER Parallel SP/1 introduced. Although the SP architecture is not included in the evaluated configuration, it uses RS/6000 hardware running AIX. • AIX Version 3.2.5 released.
1994	• AIX Version 4.1. Version 4 included major changes for thread-based scheduling, enhancements to JFS, and System V streams. Version 4 also included changes to comply with industry standards such as POSIX and XPG4. • RS/6000 SMP hardware introduced.
1996	• AIX Version 4.2. Greater than 2GB (up to 64GB) file support introduced. ITSEC E3/F-C2 security certification (completed 1997). Extensive standards compliance, including UNIX 95 branding.
1997	• AIX Version 4.3.0 64-bit ABI and concurrent 32-bit and 64-bit application support. Concurrent IP-V4 and IP-V6 support. Web-based System Management (WSM). ITSEC E3/F-C2 security certification (completed April 1998). HTML based documentation. Print spooler enhancements to handle up to 1000 jobs. • Introduction of following RS/6000 hardware platforms used in evaluated configuration: RS/6000 43P (workstation) RS/6000 Model F50 (workgroup server) RS/6000 Model S70 (enterprise server).
1998	• AIX Version 4.3.1 announced and general availability scheduled. See Announcement Letter No. 298-108. • Continuing hardware evolution, including 43P Model 150 and new Ethernet and Token Ring adapters. • AIX 4.3.1 is UNIX98 Branded. • AIX 4.3.1 is POSIX 1003.1c compliant. • RS/6000 Distributed System running AIX Version 4.3.1 TCSEC Evaluated C2 Security completes C2 evaluation.
1999	• AIX Version 4.3.1 TCSEC Evaluated C2 Security announced and generally available.

2. SYSTEM OVERVIEW

The Target of Evaluation is a system of IBM RS/6000 host computers connected via a physically protected Local Area Network. The hosts that make up the distributed system communicate using the TCP/IP protocol.

The RS/6000 is a line of high performance Uni-Processor and Symmetric Multi-Processor computers based on 32-bit and 64-bit PowerPC processors. All hosts run the same C2 version (AIX Version 4.3.1 TCSEC Evaluated C2 Security) of the Advanced Interactive eXecutive (AIX) 4.3.1 operating system. AIX Version 4.3.1 TCSEC Evaluated C2 Security is a general-purpose time-sharing operating system based on AT&T System V UNIX, incorporating many of the network functions and other enhancements from Berkeley Software Distribution UNIX. AIX is differentiated from other UNIX products by its system administration tools, Journaled File System, ability to page out portions of the kernel, preempt the kernel, and dynamically load kernel extensions.

2.1 Typical Environment and Applications

The Target of Evaluation (ToE) consists of one or multiple interconnected RS/6000 computer(s). Each of these computers is running AIX Version 4.3.1 TCSEC Evaluated C2 Security. Various models of RS/6000 computers are included in the evaluated configuration. Some are characterized as workstations and others as servers.

This closed network of hosts operates as a single system in the following aspects:

- All hosts run AIX Version 4.3.1 TCSEC Evaluated C2 Security.
- All hosts share a common user and group database. The user and group files are maintained on a master server and exported to all hosts via the Network File System (NFS). The shared information includes user name to User ID (UID) mappings, group name to Group ID (GID) mappings, user security attributes, and user authentication data. There may only be one master server per system configuration, even if that system consists of multiple LAN segments.
- All hosts have a common mapping of host names to Internet Protocol (IP) addresses.
- Administration is centrally and uniformly applied to all hosts.
- Audit tools are available and are used to merge audit files to assist in the tracking of an individual's activities across multiple host computers.

Local Area Networks (LANs) are included within the evaluation boundary because they are the equivalent of a backplane bus within a multiprocessor computer. No external routers, bridges, or repeaters are included in the configuration. When a system includes more than one LAN segment, AIX provides kernel-based routing of IP from one network to another. Figure 2-1 illustrates three compliant configurations: (a) a single, standalone RS/6000 system, (b) multiple RS/6000 computers on a LAN, and (c) a system with multiple LAN segments of the evaluated RS/6000 Distributed System.

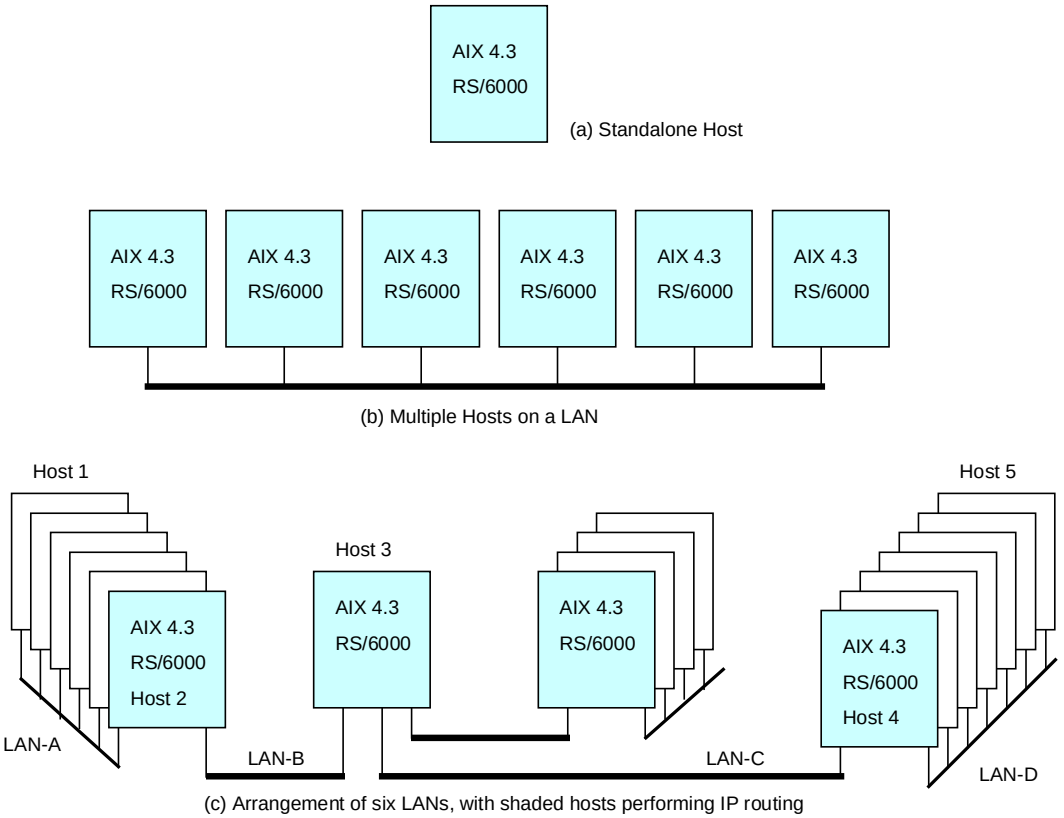

Figure 2-1. RS/6000 System Configurations. *The RS/6000 Distributed System may be configured as a single, standalone RS/6000 system, multiple RS/6000 computers on a LAN, or as a system with multiple LAN segments.*

2.2 High-Level Overview

The RS/6000 Distributed System provides a multi-user, multitasking environment, where users interact with the operating system through commands issued to a command interpreter. The command interpreter invokes command programs, which in turn function by making system calls to the operating system kernel. The Trusted Computing Base (TCB) is comprised of the kernel and trusted processes (trusted programs that are not part of the kernel). All operations performed by users are mediated by the TCB in accordance with the policies defined in Chapter 7.

Within the RS/6000 Distributed System a user can LOGIN to the console of any RS/6000 host computer, request local services at that computer, as well as request network services from any other host in the system.

Processes perform all activity. A process may be started by a user issuing a command, may be created automatically to service a network request, or may be part of the running system created at system initialization. Each process is running a program. A process may begin running a new program (i.e., via the ***exec*** system call), or create a copy of itself (i.e., via the ***fork*** system call). Some activities, such as responding to network requests, are performed directly by the kernel.

The following sections discuss services provided by the kernel, by non-kernel trusted software, and the network services. Network services are discussed separately because their implementation is split between kernel and non-kernel components.

2.2.1 Kernel Services

The AIX kernel includes the base kernel and kernel extensions. The base kernel includes support for system initialization, memory management, file and I/O management, process control, audit services and Inter-Process Communications (IPC) services. Kernel extensions and device drivers are separate kernel software modules that perform specific functions within the operating system. Device drivers are implemented as kernel extensions.

The base kernel has the following key characteristics:

- **Can be paged out:** Portions of the kernel code and data can be paged out, permitting the kernel to run using less memory than would be required for the whole kernel.
- **Pinned:** Part of the kernel is always resident or "pinned" into memory and cannot be paged. Pinned code cannot call kernel services that may result in a page fault.
- **Can be preempted:** The AIX kernel can be preempted. Higher priority threads may interrupt the kernel thread, providing support for time critical functions.
- **Dynamic and extendible:** In standard AIX, kernel extensions can be loaded and unloaded while the system is running to allow a dynamic, extendible kernel without requiring a rebuild and reboot. In the evaluated configuration, dynamic changes to the kernel are prohibited through TFM warnings. At system start up, only the kernel extensions that are part of the evaluated product may be loaded. As an example, the administrator can add pieces of evaluated hardware to a specific configuration and reboot the system. This will cause the kernel extensions that support the needed device drivers for the new hardware to be loaded. The ability to load/unload kernel extensions is restricted to the root identity.

The AIX kernel implements a virtual memory manager (VMM) that allocates a large, contiguous address space to each process running on the system. This address space is spread across physical memory and paging space on a secondary storage device. The VMM manages the paging space used by the AIX file system and provides memory buffers for use within the kernel. The file system and VMM are tightly coupled. Disk pages, whether for file I/O or paging space, are faulted into free pages in memory. The VMM does not maintain a separate pool of pages solely for file system I/O.

The process management component includes the software that is responsible for creating, scheduling, and terminating processes and process threads. Process management allows multiple processes to exist simultaneously on a computer and to share usage of the computer’s processor(s). A process is defined as a program in execution, that is, it consists of the program and the execution state of the program.

Process management also provides services such as inter-process communications (IPC) and event notification. The base kernel implements

- named pipes
- unnamed pipes
- signals
- System V semaphores
- System V shared memory
- System V message queues
- Internet domain sockets
- UNIX domain sockets
- Audit event generation

The file and I/O software provides access to files and devices. The AIX Logical File System (LFS) provides a consistent view of multiple physical file system implementations. There are four different types of file systems included in the evaluated configuration: Journaled File System (JFS), CDROM File System (CDRFS), Network File System (NFS) and the Special File File System (SPECFS). JFS and CDRFS work off of a physical medium (disk, CDROM) and NFS works across the network. SPECFS is a file system used internally by the kernel to support disk and other physical and virtual device I/O.

2.2.2 Non-Kernel Services

The non-kernel TCB services are:

- Batch processing using AT and CRONTAB
- Printer services
- Tape services (for administrator use only)
- AIX Administration (WSM)
- Identification and Authentication services
- Auditing journaling and post-processing services
- Network application layer services

Full descriptions of these services are provided in Chapter 5.

2.2.3 Network Services

Each host computer in the system is capable of providing the following types of services:

- Local services to the user currently logged in to the local computer console.
- Local services to previous users via deferred jobs.
- Local services to users who have accessed the local host via the network using protocols such as telnet.
- Network services to clients on either the local host or on remote hosts.

Network services are provided to clients via a client-server architecture. This client-server architecture refers to the division of the software that provides a service into a client portion, which makes requests, and a server portion, which carries out client requests (usually on a different computer). A service protocol acts as the interface between the client and server.

The primary low-level protocols are Internet Protocol (IP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). IP is not user visible, but non-TCB processes may communicate with other hosts in the distributed system using a reliable byte stream or unreliable datagrams, TCP and UDP respectively.

The higher-level network services are built on TCP or UDP. While the TCB supports the TCP application protocols listed below, only the timed application protocol uses UDP and is provided by the TCB in the evaluated configuration. The application protocols provided using TCP are:

- Internet remote login and file transfer services (telnet and ftp) are supported within the evaluated product, as are similar BSD interfaces, including remote command execution (rlogin, rcp, rsh, rexec).
- The Simple Mail Transfer Protocol (SMTP) is supported for mail delivery across the distributed system.
- The lpd protocol is supported for remote printing.
- The Hyper-Text Transfer Protocol (HTTP) is used by the WebInfo document display system (docsearch) for the presentation of public data.
- The WSM protocol is supported for remote host administration.
- The Network File System (NFS) protocol is supported for remote file access. This includes some subsidiary protocols, such as the Remote Procedure Call (RPC), portmap protocols, and the mountd protocol for file system import and export.

The RS/6000 Distributed System includes multiple X Windows clients in addition to an X Windows server on each host. Each server accepts connections from local clients using UNIX domain sockets.

2.3 Security Policy

Since the ToE is distributed across multiple host computers, each running a semiautonomous instance of the C2 version of the AIX operating system, the policy is described as follows:

- There is not a single kernel; rather, there is an AIX kernel running on each host computer in the system.
- The system does not have a common memory space; rather, each host in the system has its own memory space. Memory management, segmentation and paging are all managed locally, without respect to other hosts.
- Identification and authentication (I&A) is performed locally by each host computer, but uses a common database. Each user is required to LOGIN with a valid password and user identifier combination at the local workstation and also at any remote computer where the user can enter commands to a shell program (e.g., remote login, and telnet sessions).

- Neither the process ID, nor the associated thread IDs, are unique within the system; rather, a PID, and its associated TIDs, are unique on each host within the system. Process and thread management is performed locally, without respect to other hosts.
- The names of objects may not be unique within the system; rather, object names are unique on each host. For example, each host maintains its own local file system, but may mount NFS exported file systems at various locations in the local directory tree.
- Discretionary access control (DAC) is performed locally by each of the host computers and is based on user identity and group membership. Each process has an identity (the user on whose behalf it is operating) and belongs to one or more groups. All named objects have an owning user, an owning group and a DAC attribute, which is a set of permission bits. In addition, file system objects optionally have an extended permission list also known as an Access Control List (ACL). The extended permissions mechanism is a significant enhancement beyond traditional UNIX systems, and permits control of access based on lists of users and/or groups to whom specific permissions may be individually granted or denied.
- Object reuse is performed locally, without respect to other hosts.
- Audit is performed locally by each host computer. The audit facility generates audit records for activities performed directly by untrusted processes (e.g., the system calls that perform file I/O) as well as trusted process activities (e.g., requests for batch jobs). Audit tools are available to merge audit files from the various hosts.
- Interrupt handling is performed locally, without respect to other hosts.
- Privilege is based on the root identity. All privileged processes (setuid root programs and programs run under the root identity) start as processes with all privileges enabled. Unprivileged processes, which include setgid trusted processes, start and end with no privileges enabled.

2.4 TCB Structure

The Trusted Computing Base is the portion of the system that is responsible for enforcing the system's security policy. The RS/6000 Distributed System TCB is distributed across each RS/6000 host computer and consists of four major components: hardware, kernel software, kernel extension software, and trusted processes. All these components must operate correctly for the system to be trusted.

The RS/6000 Distributed System hardware components support two execution states where kernel mode or supervisor state, software runs with hardware privilege and user mode or problem state software runs without hardware privilege. AIX also provides two types of memory protection: segmentation and page protection. The memory protection features isolate critical parts of the kernel from user processes and ensure that segments in use by one process are not available to other processes. The two-state architecture and the memory protections form the basis of the argument for TCB and process isolation.

The trusted processes include programs such as AIX administrative programs, scripts, shells, and standard UNIX utilities that run with administrative privilege, as a consequence of being invoked by a user with the root identity. Non-kernel TCB software also includes daemons that provide

system services, such as networking and managing audit data, as well as setuid and setgid programs that can be executed by untrusted users.

2.5 TCB Interfaces

Each sub-section here summarizes a class of interfaces in the RS/6000 Distributed System, and characterizes them in terms of the TCB boundary. The TCB boundary includes some interfaces, such as commands implemented by privileged processes, which are similar in style to other interfaces that are not part of the TCB boundary and thus not trusted. Some interfaces are part of the TCB boundary only when used in a privileged environment, such as an administrator's process, but not when used in a non-privileged environment, such as a normal user process. All interface classes are described in further detail in the next chapter, and the mechanisms in subsequent chapters. As this is only an introduction, no explicit forward references are provided.

2.5.1 Untrusted User Interfaces

The typical interface presented to a user is the command interpreter, or shell. The user types commands to the interpreter, and in turn, the interpreter invokes programs. The programs execute hardware instructions and invoke the kernel to perform services, such as file access or I/O to the user's terminal. A program may also invoke other programs, or request services using an IPC mechanism. Before using the command interpreter, a user must log in.

The TCB interfaces presented to the user are:

- **CPU instructions**, which a process uses to perform computations within the processor's registers and a process's memory areas;
- **System calls** (e.g. ***open, fork***), through which a process requests services from the kernel, and are invoked using a special CPU instruction;
- **Directly-invoked trusted processes** (e.g. **ENQ**) which perform higher-level services, and are invoked with an ***exec*** system call that names an appropriate program in the TCB, and replaces the current process's content with it;
- **Daemons** (e.g. **cron**), which accept requests stored in files or communicated via other IPC mechanisms, generally created through use of directly invoked trusted processes.
- **Distributed Services**, (e.g. telnet**,** NFS**,** rsh) The distributed services interface operates at many different levels of abstraction. At the highest level, it provides a means for users on one host to request a virtual terminal connection on another host within the system. At a lower level, it allows a host on the distributed system to request a specific service from another host within the system on behalf of a user. Examples of requested services include, reading data from a designated file (i.e. NFS), executing a command line (e.g. rsh), transfering whole files (e.g. FTP), or delivering mail (i.e. SMTP). At the lowest level, it allows a subject on one host in the system to request a connection (i.e. TCP), or deliver data (i.e. UDP) to a listening subject. All the distributed interfaces are TCB interfaces, although, as noted above, some simply establish communications with a non-TCB subject.

2.5.2 Operation and Administrator Interface

The primary administrative interfaces to the RS/6000 Distributed System are the same as the interfaces for ordinary users; the administrator logs into the system with a standard, untrusted, identity and password, and after assuming the root identity uses standard AIX commands to perform administrative tasks.

The RS/6000 Distributed System is composed of one or more 43P, F50 or S70 RS/6000 computer systems. Each of these host computers may be in one of the following states: shut down, initialization, single-user mode, or multi-user secure state. Administration entails the configuration of multiple computers and the interactions of those computers, as well as the administration of users, groups, files, printers, and other resources within the system.

The RS/6000 Distributed System provides two general purposes, menu-based utilities for system administration: WSM and SMITTY. Other programs (e.g., **/USR/BIN/ACLEDIT, /USR/BIN/CHUSER, /USR/BIN/RM**) and scripts are used for system administration, but WSM and SMITTY are significant because they provide comprehensive system administration capabilities.

WSM and SMITTY are required for the administration of the RS/6000 Distributed System, but the decision as to which administrative utility to use depends upon whether or not the system is in a secure state:

- SMITTY (a cursor-based ASCII version of the System Management Interface Tool (SMIT)) is a graphical interface and dispatcher for a collection of administrative programs. SMITTY is used to administer the local host, i.e., the computer where it is run. It may be used only while the computer is running in single-user mode (for example, installation or error recovery). The basis for this limitation on the use of SMITTY is described in the Trusted Facility Manual (TFM).
- WSM is IBM's new-generation administrative architecture. WSM is implemented through a client-server architecture, where the client software runs on the administrator's local host and the server runs on either the administrator's local host or any other computer in the system. The WSM server is a dispatcher for a collection of administrative programs, many of which are the same as the programs invoked by SMITTY. WSM is included in the TCB and, thus, may be used to administer a host while the system is in a secure state.

The system maintains an administrative database on a Network File System (NFS) server, referred to as the administrative master server. The remaining hosts import the administrative data from the master server through ordinary NFS client operations. Once the system is running in a multi-user secure state, the Trusted Facility Manual requires that only Web-based System Management (WSM) be used for system administration.

2.5.3 Secure and Non-Secure States

The secure state for the RS/6000 Distributed System is defined as a host's entry into multi-user mode with auditing fully operational, and with the administrative databases NFS-mounted from the master server. At this point, the host accepts user logins and services network requests across the distributed system. If these facilities are not available, the host is considered to be in a non-

secure state. Although it may be operational in a limited sense and available for an administrative user to perform system repair, maintenance, and diagnostic activity, the TCB is not in full operation and is not necessarily protecting all system resources according to the security policy.

3. TCB INTERFACES

This chapter describes the interfaces that together comprise the RS/6000 Distributed System TCB interface.

It should be noted that not all the interfaces would necessarily be characterized as TCB interfaces in the evaluated product. The following will exclude an interface as a TCB interface:

- Interfaces between elements of the TCB are not categorized as TCB interfaces unless untrusted users or software can also invoke them.
- Calls to library functions are not TCB interfaces, because the library function is either statically or dynamically linked with the user program, and runs as part of the same process. Library functions may invoke system calls (which are TCB interfaces), or may be linked with TCB programs (which may provide TCB interfaces), but the library function is not itself a TCB interface.
- Some commands may perform TCB operations by simply invoking a system call that performs the actual work. For example, a program may invoke the ***creat*** system call to make a new file. In this case, it is the ***creat*** call, and not the program invoking it, which provides the TCB interface.

3.1 User Interfaces

The user interfaces were introduced in the previous chapter. These interfaces are used directly by system users. This section describes the interfaces in more detail.

3.1.1 Hardware Instructions

Processor instructions, described in more detail in Chapter 4, are the basic computational mechanism available to processes on the RS/6000 Distributed System. Each process only has access to its own set of explicitly addressable hardware registers, which are managed, by the kernel.

3.1.2 System Calls

System calls are the mechanism for invoking kernel services and, through system calls naming specific trusted processes or protocols, invoking all other interfaces available to local subjects. In the AIX operating system, a system call is a routine that crosses the protection boundary between the user process and kernel domains. System calls provide user-mode access to special kernel functions. Because system calls run in a more privileged protection domain they can access data that users cannot. The system call interface is the exception handler for system calls. The system call handler, running in kernel-mode, calls the kernel internal function supporting the system call. The system call function returns to the system call handler when it has performed its operation. The system call handler then restores the state of the process and returns to the user program.

All system calls are processed entirely within the kernel TCB software, isolating them from external interference. System call parameters are copied by the initial system call handler, and large data structures are copied between process memory and the kernel through routines that ensure that all memory addressed is within the process's assigned memory areas.

3.1.3 Directly Invoked Trusted Processes

A third class of TCB user interfaces exists between users and trusted processes. A trusted process program is distinguished from other programs in that its program file is part of the TCB and designated to acquire a characteristic (either the privileged identity root or a privileged group identity such as audit) causing it to run on behalf of the TCB. For example, users changes their passwords by running the **PASSWD** utility which involves a setuid program (trusted process) making controlled changes to a TCB data file.

A process invokes a trusted process through the ***exec*** system call. Typically this is done by the user's command interpreter process (e.g. **KSH**) in response to a command issued by users or a script. The ***exec*** system call names a program file for the process to be invoked.

When a trusted process executes, it is isolated from interference by the invoking user through one or more of the following kernel mechanisms:

- the process may completely change identity and prevent the invoking user from sending unwanted signals,
- the invoking user cannot trace or modify its execution using debugging services, or
- the process may set signal handlers and masks during critical regions.

3.1.4 Network Interfaces

The IBM RS/6000 Distributed System ToE is a multi-host distributed system, where each host runs AIX Version 4.3.1 TCSEC Evaluated C2 Security, and a particular user ID refers to the same user on each host. Each user initially logs in to a local host. (i.e., the computer that is directly attached to the keyboard and display used by the user.) After the user logs in to the local host, the user can run client programs that issue network requests, via the TCP/IP protocol stack, to server processes on other hosts of the system.

TCP/IP runs as a kernel extension on each host in the system. Each host also contains TCB kernel code (e.g., for NFS) and TCB daemons (e.g., ftpd, rshd, telnetd). All hosts implement the Berkeley socket system calls, (e.g., ***socket, bind, connect)***.

An IP datagram contains a source address, a destination address, an Internet protocol type, and data that is part of a higher-level protocol. Since this is a closed distributed system, network datagrams are only received from other AIX C2 hosts. The administrator is responsible for correctly assigning IP addresses and for not connecting the distributed system to other networks.

Typically, an IP datagram contains either a TCP segment or a UDP datagram; a third type, the Internet Control Message Protocol (ICMP) message, is used for TCB control functions. Both TCP and UDP identify a source port and destination port, and identify a unique subject-to-subject

communication path based on the source and destination address and port. Port numbers less than 1024 are privileged ports; a privileged source port indicates a client in the TCB, and a privileged destination port indicates a server in the TCB. In addition to ports below 1024, the RS/6000 Distributed System has the capability, through a port protection mechanism described in Chapter 5, to designate other ports as providing TCB servers and clients. The services supported by this port protection mechanism are X Windows, WSM, and NFS.

There are no interfaces between the untrusted client process and the remote host's hardware or kernel. The two host kernels act as a data pipe with respect to transferring client requests and server responses, and the untrusted user has access only to the local host's end of the pipe.

The interface between the remote host's server daemon and the remote host's kernel, is considered internal to the TCB since both are elements of the TCB.

Additional TCB interfaces consist of hardware interfaces to the adapter, software interfaces to the local driver and TCP/IP stack, and client-server interfaces to the remote server daemon.

- **Hardware interfaces to the adapter:** There are four network adapters available for use in the evaluated configuration. There are no user addressable buffers or caches included on any of these network adapters. These interfaces are considered internal to the TCB.
- **Software interfaces to the local driver:** The device drivers for each network adapter contain transmit and receive queues that are used to temporarily buffer data before it is transmitted over the wire. These queues are internal to kernel, and are not accessible by untrusted users. These interfaces are considered internal to the TCB.
- **Socket system calls**: The Berkeley socket system calls, (e.g., ***socket, bind***, ***connect***) act as the user interface for network requests. These are classified under the System Call class of TCB interfaces.
- **Client-Server interfaces**. These include the service protocol messages from the client process on the local host to the server process on either a remote or local host. These messages are typically defined in the Request for Comments (RFC) specifications for the particular protocol. All of these are considered TCB interfaces. The TCB interfaces presented by WSM, however, are covered as trusted processes and are not explicitly listed as Network TCB Interfaces.

3.1.4.1 Client-Server Interfaces

The application-level interfaces constitute the majority of user-visible network functions. In all cases the RS/6000 Distributed System can act as either a server or a client. The RS/6000 Distributed System implements the protocols according to their specifications (RFCs), and guarantees that the service is present on the appropriate port number.

Table 3.1 lists the network application and the function they provide. Details on these protocols can be found in Chapter 5, Network Applications.

Table 3-1. Network Applications. *The RS/6000 Distributed System provides network functions through network application protocols.*

Protocol	Function
telnet	remote login (virtual terminal)
ftp	remote file transfer
smtp	mail delivery
portmap	map RPC protocol and version numbers to TCP ports
rexec	remote command-line execution
rlogin	remote login (virtual terminal)
rsh	remote command-line execution
rcp	remote file transfer
lpd	printer job queuing and transfer
NFS	remote file access
WSM	administrative server

3.1.4.2 X Windows Interface

The IBM RS/6000 Distributed System includes various X Windows clients and an X Windows server on each host. Each server accepts connections from a client via a UNIX domain socket.

The X server directly manages the video display, keyboard, and mouse and runs as the user who invoked it. It is not setuid/setgid and is not privileged even when invoked by an administrator. For this reason the X server is not considered a TCB interface, as it is always either completely external to the TCB (when invoked by an untrusted user), or completely internal to the TCB (when invoked by a trusted user). Further details on the X Windows implementation within AIX are provided in Chapter 5.

3.2 Administrative Interface

WSM is a Java-based point and click facility, which acts as the administrative command interface and executes commands selected by the administrator. In order to perform administrative functions, users must successfully assume the root identity. Non-TCB subjects protect administrative databases from manipulation by way of normal file system access control. Where access is restricted to owner only, and the owner is one of the restricted ids named in table 6-2. The administrative commands rely on both kernel policy enforcement and the use of the root identity to permit the overriding of policies normally enforced by the kernel interface.

4. TCB HARDWARE

This chapter describes the security-relevant aspects of the hardware in the evaluated configuration. It describes the CPU architecture, system board components, peripherals, hardware components role in the security policy, hardware input/output, multiprocessing, memory (architecture and protection), context switching and hardware equivalence.

The components of the RS/6000 Distributed System must be protected from physical tampering as stated in the RS/6000 Distributed System TFM.

- All of the computers in the system are physically or procedurally protected from tampering and hardware modifications (replacing EPROM chips or installing bus-monitoring devices).
- The network wiring is physically or procedurally protected from wiretapping. This includes both passive wiretapping, where an intruder monitors data (perhaps passwords) passed in clear text across the network, and active wiretapping, where an intruder inserts arbitrary data onto the wire.
- The C2 system will not be connected to any non-TCB computers or computer networks. In particular, do not connect the system to the Internet.

4.1 CPU Architecture

The computers in the evaluated configuration use the PowerPC 604e, a 32-bit processor, or the PowerPC RS64, a 64-bit processor. There are two separate kernels included with AIX. One kernel is for uni-processor systems and one is for multi-processor systems. Both kernels execute in 32-bit mode on either processor. 32-bit mode refers to the fact that the CPU uses 32-bit effective addresses for memory accesses, and instructions and registers are 32-bits long. The Machine Status Register (MSR) on the 64-bit processor contains a bit that determines whether the CPU is operating in 32-bit or 64-bit mode.

The main advantages inherent with the 64-bit processor running in 64-bit mode are large file, memory, and application virtual address spaces, and 64-bit integer computation, using 64-bit general-purpose registers. These advantages are used through the development of applications that execute as 64-bit processes.

Both processors are Common Hardware Reference Platform (CHRP) compliant - they are similar to each other in design and function and adhere to a published standard. CHRP is an open systems specification for building PowerPC-based computer systems. The specification defines the devices, interfaces, and data formats that a CHRP-compliant platform must make visible to software. It also describes the services that must be provided by firmware and hardware before the operating system gains control. The CHRP document that describes the features a system must provide to be CHRP compliant is *PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture.*

4.1.1 Execution States

The processors in the evaluated configuration implement a two-state architecture, where kernel mode software runs with hardware privilege (Supervisor State) and user mode software runs without hardware privilege (Problem State). These are the only two execution states provided by these CPUs.

In user mode, a process executes application code while the machine is in a non-privileged state. The kernel data and global data structures are protected from access and modification. In kernel mode, processes exist only in the kernel protection domain and run with hardware privilege. The detailed description of the system's memory protection mechanisms is presented in section 4.6.4, Memory Protection.

The two-state architecture is enforced in hardware for every instruction and memory access. An exception is generated if a user process attempts to execute a privileged instruction or access protected memory. The distinction between user and kernel mode software is determined by the Supervisor bit, set in the MSR. This bit can only be set by privileged software, so it is not possible for user mode software to set the bit and give itself hardware privilege.

Table 4-1. Problem and Supervisor State Comparison.

Problem state	Supervisor state
User programs and applications run in this mode	The base kernel, kernel extensions, and device drivers run in this mode
Kernel data and structures are protected from access/modification	Can access/modify anything
Access to only user instructions	Access to user and privileged instructions

There are three ways for a processor to transition from problem state to supervisor state: the execution of the system call instruction, an exception, or an interrupt. Interrupts and exceptions are discussed in section 4.1.3.

A user process executes the system call instruction to request services from the kernel. This is the programming interface for user mode software to make requests of the kernel. When the system call instruction is executed, a system call exception is generated. The processor switches to Supervisor State to handle the exception. The effective address of the instruction following the system call instruction is placed into the Save Restore Register 0 (SRR0). A collection of bits from the MSR are placed into the corresponding bits of Save Restore Register 1 (SRR1). The system call instruction passes a pointer to the system call table that identifies the system call to be processed.

A description of the specific information that is saved and restored when a context switch occurs can be found in section 4.7, Context Switching.

4.1.2 CPU Registers

The following section describes three sets of CPU registers. The first set is composed of the program-visible registers that an unprivileged program can manipulate through machine instructions. The second set is visible to both user and kernel mode software, but can only be written by the TCB while in kernel mode. The third set is accessible only by the TCB in kernel mode and is used for system control functions.

The registers integrated into the PowerPC 604e are 32-bits long, with the exception of the floating-point registers, which are 64-bits long. Most of the registers integrated into the PowerPC RS64 chip are 64-bits long. Tables 4-2, 4-3, and 4-4 list the CPU registers that are contained within each processor and their width.

When a context switch occurs, the currently executing thread on the system stores a subset of the Problem State and Supervisor State registers. This allows the unique register values of a particular thread to be restored when the thread next executes. Context Switching is discussed in detail in section 4.7.

When 64-bit registers are used in 32-bit mode, the high-order 32-bits of the effective address are ignored when accessing data and are set to zero when executing instructions. This eliminates the problem of residual data being maintained in the register before or after a 32-bit operation is performed on a 64-bit register. The high order bits of the 64-bit registers are not saved or restored during a context switch, when the processor is operating in 32-bit mode.

4.1.2.1 Problem State Registers

Problem state registers are those registers that can be read and manipulated by user mode software.

Table 4-2. Problem State Registers.

Register	Description	32-bit CPU	64-bit CPU
GPRs	General Purpose Registers, can hold integer or address, used for loads, stores, and integer calculations	(32) 32-bit	(32) 64-bit
FPRs	Floating Point Registers, used in floating point operations	(32) 64-bit	(32) 64-bit
CR	Condition Register, contains bits set by the results of compare instructions	(1) 32-bit	(1) 32-bit
FPSCR	Floating-Point Status and Control Register, contains exception signal bits, exception summary bits, exception enable bits, and rounding control bits need for compliance with IEEE 754 (defines operations of binary floating-point arithmetic and representations of binary floating-point numbers)	(1) 32-bit	(1) 32-bit
XER	Contains Summary Overflow, Overflow, Carry flags and byte-count for string operations.	(1) 32-bit	(1) 32-bit
LR	Link Register, set by some branch instructions, points to the instruction immediately after the branch	(1) 32-bit	(1) 64-bit
CTR	Count Register, can be decremented, tested, and used to decide whether to take a branch, all from within one instruction	(1) 32-bit	(1) 64-bit

4.1.2.2 Shared Registers

Shared registers are those registers that are read-only for user mode software, and read/write for the kernel. The Time Base Facility (TB) is a pair of registers that are shared between problem and supervisor state. The TB registers maintain the time of day and operate interval timers. User mode software may query the TB registers, but only the kernel may affect their values.

Table 4-3. Shared Registers.

Register	Description	32-bit CPU	64-bit CPU
TB	Time Base Facility (TBL and TBU), operates an interval timer	(2) 32-bit	(2) 32-bit

4.1.2.3 Supervisor State Registers

Supervisor state registers are those registers that are only accessible by the kernel.

Table 4-4. Supervisor State Registers.

Register	Description	32-bit CPU	64-bit CPU
MSR	Machine State Register, controls many of the operating characteristics of the processor (privilege level: supervisor vs. problem, addressing mode: real vs. virtual and interrupt enabling)	(1) 32-bit	(1) 64-bit
PVR	Processor Version Register, identifies the version and revision level of the PowerPC processor (read-only)	(1) 32-bit	(1) 32-bit
IBAT	Instruction BAT Registers, maintain the address translation for 4 blocks of memory	(8) 32-bit	(8) 64-bit
DBAT	Data BAT Registers, maintain the address translation for 4 blocks of memory	(8) 32-bit	(8) 64-bit
SDR1	Contains control information for page table structure	(1) 32-bit	(1) 64-bit
ASR	Address Space Register, points to the segment table	none	(1) 64-bit
SR	Segment Registers, used in virtual addressing mode to provide a large virtual address space, simulated in 64-bit mode	(16) 32-bit	(16) 32-bit (emulated using the 32-bit bridge facilities, see 4.7.1.2.2)
DAR	Data Address Register, contains the memory address that caused the last memory related exception	(1) 32-bit	(1) 64-bit
DSISR	Identifies the cause of DSI and alignment exceptions	(1) 32-bit	(1) 32-bit
SPRs	Special Purpose Registers - general operating system use	(4) 32-bit	(4) 64-bit
SRR	Machine State Save/Restore Register, these registers save information when an interrupt occurs, SRR0 points to the instruction that was running when the interrupt occurred, and SRR1 contains the contents of the ***MSR*** when the interrupt occurred	(2) 32-bit	(2) 64-bit
DABR	Data Address Breakpoint Register, detects access to a designated double-word	(1) 32-bit	(1) 64-bit
DEC	Decrementer, used for programmable delays	(1) 32-bit	(1) 32-bit
EAR	External Address Register, used to identify the target device for external control operations	(1) 32-bit	(1) 32-bit
IAR	Instruction Address Register, or the program counter	(1) 32-bit	(1) 64-bit

4.1.3 Interrupts and Exceptions

From a hardware perspective, a PowerPC CPU treats all interrupts to the processing flow as hardware exceptions. One specific exception is used for all device interrupts. The purpose of interrupts and exceptions is to allow the processor to enter Supervisor State as a result of external signals, errors or unusual conditions arising from the execution of an instruction.

4.1.3.1 Interrupts

An interrupt is an asynchronous event not associated with the instruction being executed at the time of the interrupt. An interrupt causes the kernel to save the thread state, execute the interrupt handler, and eventually restore and restart the thread that was executing.

All I/O interrupts originate with a device generating an electrical signal to the hardware interrupt controller on the I/O planar. The interrupt controller is programmed by the kernel to associate the interrupt levels generated by the hardware devices with the corresponding software priority levels used by AIX, where the interrupt priority is determined by the characteristics of the device.

The kernel maintains a set of structures to store context when an interrupt occurs. These structures contain fields for all of the register values that are saved when an interrupt occurs. Each processor in the system has a pointer to the structure it will use when the next interrupt occurs. This pointer is called the Current Save Area (CSA).

When an interrupt occurs, the kernel saves the following registers into the CSA: instruction address, machine status, condition, link, count, segment registers, and general purpose registers.

The kernel then gets the next available structure, links the structure that was just used to the new structure, and updates the CSA for the processor. Interrupts each have a priority, so higher priority interrupts can occur while a lower priority interrupt handler is executing. This results in additional structures being allocated and linked together.

Table 4-5. MSR high-order bits. *The machine status register contains the current state of the CPU in bits 16-31 on a 32-bit processor or bits 48-63 on a 64-bit processor.*

Name	Description	32-bit index	64-bit index
EE	External Interrupt Enable	16	48
PR	Privilege Level	17	49
FP	Floating-point Available	18	50
ME	Machine check enable	19	51
FE0	Floating-point Exception Mode 0	20	52
SE	Single step trace enable	21	53
BE	Branch trace enable	22	54
FE1	Floating point Exception Mode 1	23	55
IP	Exception Prefix	25	57
IR	Instruction address translation	26	58
DR	Data address translation	27	59
RI	Recoverable exception	30	62
LE	Little endian mode enable	31	63

When an interrupt occurs, SRR0 maintains the effective address of the instruction that the processor would have attempted to access next if no interrupt condition were present. SRR1 stores bits 16-31 in 32-bit mode or bits 48-63 in 64-bit mode from the MSR, which contain the state of the processor at the time of the interrupt.

4.1.3.2 Exceptions

An exception is a synchronous event directly caused by the instruction being executed when the exception occurs; where an exception is any condition that changes the flow of processing. Exceptions include synchronous events caused by user programs (program errors and undefined instructions) and system conditions, such as page faults or segment faults. The most commonly occurring exception is a page fault.

Exception priorities are organized by exception class and are in descending order of priority: asynchronous, synchronous, imprecise and mask-able. Exceptions may occur while an exception handler is executing. Servicing of exceptions may be nested due to the presence of a higher priority exception.

The evaluated CPUs provide a method of explicitly enabling or disabling particular exception conditions. This subset of exceptions is enabled and disabled using bit values stored in the MSR. The MSR can only be modified while the CPU is operating in Supervisor State, so the enabling and disabling of exceptions cannot be performed by user mode software.

When an exception occurs, the processor switches to Supervisor State and immediately branches to an exception vector. Each exception type has a defined vector that contains the memory address of the exception handler to be executed. The effective address is stored in SRR0 before the branch. The address is either the location of the next instruction to execute when the exception handler returns, or the instruction that was currently executing when the exception occurred.

The kernel determines if the process has defined a signal handler for this exception. If so, the signal handler is invoked. If no signal handler is defined, the process is terminated according to the default behavior that is specified for that exception.

4.1.4 Instruction Set Overview

This section presents problem and supervisor state instructions. Problem state instructions provide the computational, branching and storage/retrieval instructions, which can be executed in problem or supervisor state. Supervisor state instructions perform specialized functions in support of the operating system and can only be executed when the CPU is operating in Supervisor State. If a supervisor state instruction is issued from Problem State, the processor will generate a program exception. The process that occurs following a program exception is detailed in section 4.1.3.2, Exceptions.

The instruction format consists of an op-code and zero or more operands. The op-code is the instruction to be executed, and the operands are the registers to be used by the instruction, to achieve a result. The PowerPC architecture uses 4-byte instructions that are word aligned.

The problem state instructions are broken down into the following classes.

Table 4-6. Classes of Problem State Instructions.

Class	Description
Integer	Integer arithmetic, compare, logical, rotate and shift instructions
Floating point	Floating point arithmetic, multiply-add, rounding, conversion, compare, status, control register and move instructions
Load and store	Integer load, integer store, integer load and store with byte reverse, load and store multiple, floating point load, floating point store, load word and reserve indexed, store word conditional indexed, and memory synchronization instructions
Branch and flow control	Branch, condition register logical, trap instructions
System linkage	System call instruction
Processor control	Move to/from condition register, move to/from XER, LR, CTR, move from time base
Memory synchronization	Synchronize, execute in order and instruction synchronize
Memory control	User level cache instructions (these instructions are described below in section 4.1.5.1)

The supervisor state instructions are broken down into the following classes.

Table 4-7. Classes of Supervisor State Instructions.

Class	Description
System linkage	Return from interrupt
Processor control	Move to/from machine state register, move to/from ***PVR, IBAT, DBAT, SDR1, DAR, DSISR, SPRs, SRRs, DABR, EAR*** and ***IAR***
Memory control	Supervisor level cache management, segment register manipulation, and translation look-aside buffer management instructions

The Service Processor Attention instruction is a supervisor state instruction used by the AIX kernel to communicate with the S70 Service Processor. The instruction is only available on the IBM RS64 processor and is described by the following:

Table 4-8. Service Processor Instruction.

Class	Description
Service Processor Attention	When the kernel requires information from the Service Processor, it puts a request in a buffer used to transfer data between the service processor and the kernel, interrupts the Service Processor, and waits for a response from the Service Processor.

4.1.5 CPU Caching

Level 1 cache (L1) is a component of the processor, and is provided to cache recently used blocks of memory. Both the 604e and the RS64 provide separate L1 caches for instructions and data. The 604e provides separate 32 K-byte instruction and data caches, while the RS64 provides separate 64 K-byte instruction and data caches.

When a process references memory it refers to an effective address. The effective address is translated into a temporary virtual address by the memory management hardware. The virtual address is calculated by referencing the segment registers, which provide the high order bits for the virtual address. These high order bits specify the virtual segment that this address is contained

within.

When the L1 and L2 caches are referenced, the lookup is based on the virtual address. Each virtual address belongs to only one process. The kernel assigns blocks of virtual addresses (segments) to a process when the process is initialized. Section 4.6.2, Address Translation describes how address translation is performed and the significance of virtual addresses.

Each page in memory has four bits that define the pages caching attributes in the page table. These cache bits are modifiable by the VMM, and are not modifiable by user mode programs. WIMG bits are set to the default (0010) when a page is allocated. The only modification to the WIMG bits occurs prior to an I/O operation (0111). The bits are set to bypass any L1 or L2 caching while performing load and store operations to an adapters memory address range.

Table 4-9. WIMG bits. *The WIMG bits are stored in the page-table entry for a page, and control the implementation of caching for that page.*

Cache Feature	Bit = 0	Bit = 1
Write Through (W)	The processor is only required to update the cache. Main memory may be updated due to another operation. (Write back)	Store operations update the data in the cache and also write through to update the data in memory. (Write through)
Caching Inhibited (I)	The processor performs load and store operations using the cache or main memory, depending on the other cache settings. (Cache allowed)	The processor bypasses the cache and performs load and store operations to main memory. (Caching Inhibited)
Memory Coherency (M)	Store operations to that location are not serialized with all stores to that same location by all other processors that also access the location coherently. (Memory coherency not required)	Store operations to that location are serialized with all stores to that same location by all other processors that also access the location coherently. (Memory coherency required)
Guarded (G)	Allows out of order access to memory. (Not guarded)	Prevents out of order access to memory. (Guarded)

4.1.5.1 User level Cache Instructions

Table 4-10. User level cache instructions. *The following instructions allow a processor to manipulate on-chip caches, within the bounds of its virtual address space.*

Name	Mnemonic	Description
Data Cache Block Touch	dcbt	Allows a user mode program to pre-fetch cache blocks as a performance enhancement to the execution of their program, specified by the effective address.
Data Cache Block Touch for Store	dcbtst	Same as dcbt, except for store operations.
Data Cache Block Set to Zero	dcbz	Write zeros into a cache block specified by an effective address.
Data Cache Block Store	dcbst	Forces a write of a cache block to memory specified by the effective address.
Data Cache Block Flush	dcbf	Flushes a cache-block specified by the effective address, copying the block to memory if there is data in it.
Instruction Cache Block Invalidate	icbi	Invalidates the block containing the byte addressed by the effective address, so that subsequent references cause the block to be restored from real memory.

User level cache instructions provide a method for a normal user program to manipulate the cache. The following instructions are the only interfaces available to the cache for a user. There are other cache instructions available, but those instructions are Supervisor State only. User level cache instructions generate exceptions if they attempt an operation that violates the current attributes for the block of memory being referenced.

4.2 System Board Components

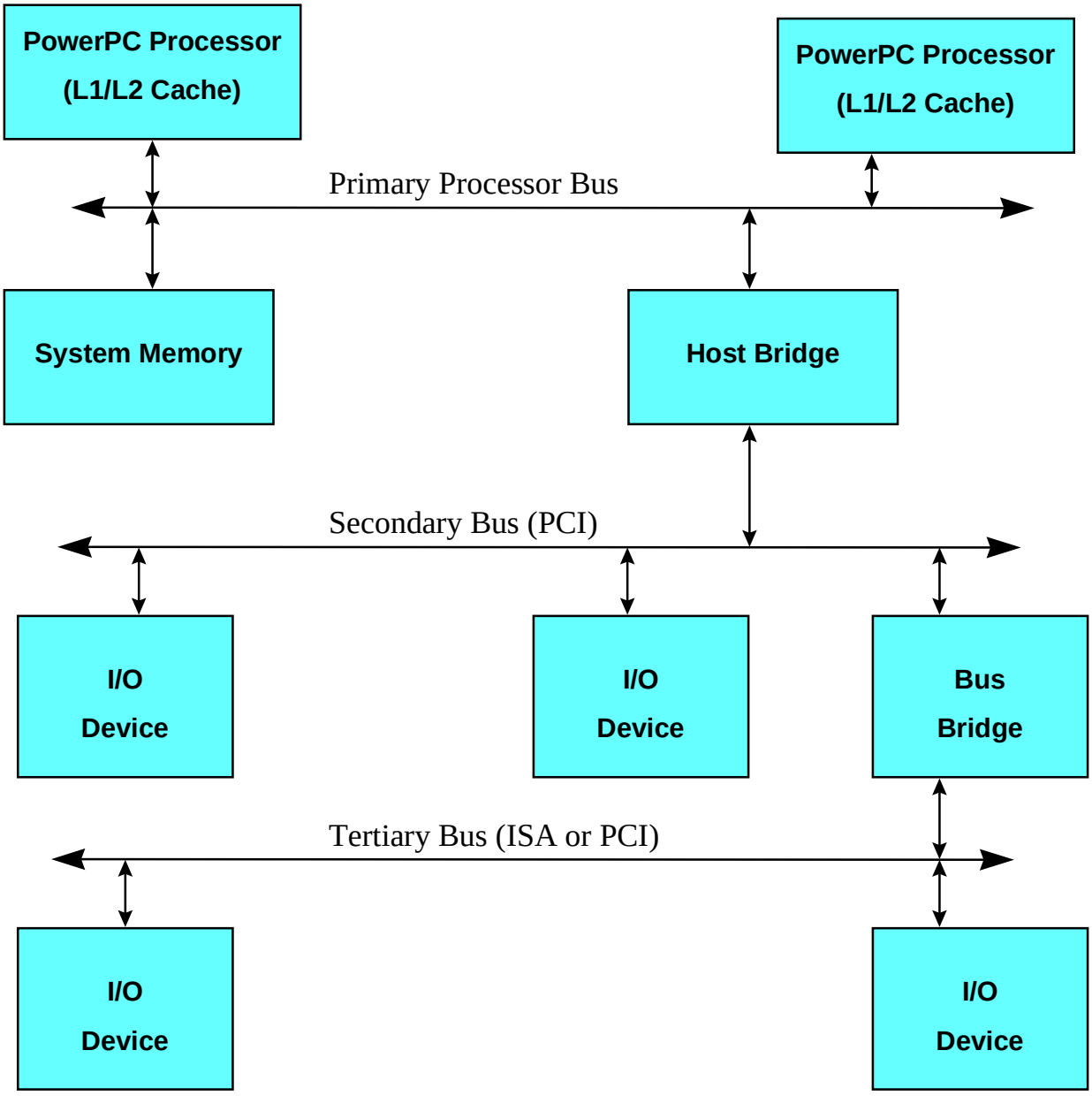

Figure 4.1: Generic Representation of a PowerPC System

This section describes the motherboard components, including the buses used in the system and the elements of the system that are contained on each motherboard. The following figure demonstrates bus connections between the PowerPC Local Bus, the PCI bus, and the ISA bus.

4.2.1 Buses and Bridges

The systems in the evaluated configuration support three different types of buses: PowerPC Local Bus, Peripheral Component Interface (PCI), and Industry Standard Architecture (ISA). The S70 supports an additional bus, Remote Input/Output (RIO).

The PowerPC Local Bus is a memory bus that directly connects the processor or processors to system memory. As this is a memory bus, the only devices that are attached are memory modules and processors or processor cards. The local bus indirectly connects the processor or processors to the PCI bus and ISA bus via host bridges on the planar.

The PCI bus is the primary I/O bus for each of the computers, used to connect high-speed peripherals to the planar. Some of these devices are implemented on the planar (SCSI-2 controller, bridge to ISA bus), while the remainder (LAN adapters, video adapters, and additional SCSI controllers) are implemented as adapter cards that plug into PCI slots on the planar.

The ISA bus is a low bandwidth bus for connection of low-speed peripherals, non-volatile RAM, audio, and power manager logic. The ISA bus supports the keyboard, mouse, parallel port, and diskette drive.

The RIO Bus is a high-speed multiplexed I/O bus that is used to connect the S70 system rack with its I/O drawers. The S70 system rack contains the CPU cards and memory chips. A separate cabinet or cabinets are provided for connection of I/O devices. The RIO bus is a high-speed extension of the PCI bus, from the system rack to the I/O drawers.

Each of the buses supported must be connected on the system planar. The chips and data paths that connect the different buses of a system are referred to as bridges.

The hardware buses do not enforce policy. They are used to transfer data between the different components of the system.

While there are no caches related to the system buses, there are buffers. These buffers are used in pipelining I/O accesses, and are implemented as small FIFOs. None of these buffers are addressable by untrusted subjects. These buffers are only manipulated using diagnostic routines. The individual FIFOs on the system board may be turned off to isolate problems internal to the main board.

4.2.2 System Board Elements

4.2.2.1 Level 2 Cache

Level 2 cache (L2) is a high-speed collection of memory, separate from L1 cache, but providing a similar function. The L2 cache contains recently accessed memory locations, and is referenced

following address translation to determine if the memory address is currently stored in the cache. L2 cache is slower to access than L1, but is faster than accessing system memory directly. If a memory location is referenced that is not in L1 or L2 cache, both L1 and L2 are updated.

The L2 cache on each of the evaluated systems is larger than the L1 cache. The 43P has 1MB of L2 cache, directly on the planar. The F50 has 256 K-byte of L2 cache per 604e processor, for a total of 1MB. The L2 cache on the F50 is not directly shared among the four processors. The S70 has four MB of L2 cache per processor. This L2 cache is not directly shared between the four processors on a CPU card.

4.2.2.2 Open Firmware and Flash Memory

Open Firmware is the logic responsible for discovering the devices that are installed in the system, passing this list of devices to the operating system, and transferring control of the hardware to the operating system. Open Firmware is stored on a system ROM on each of the planars and is defined by IEEE 1275-1994. When an RS/6000 system is powered on, Open Firmware has control.

All the systems make use of a power-on password and a privileged password as a function of system configuration. These passwords are stored and accessed while Open Firmware has control of the machine. The RS/6000 TFM states that the privileged password must be set during the installation process. The privileged access password controls access to the Open Firmware settings for the machine. The power-on password is an optional feature for restricting the ability to power on the machine. If the machine is powered off and back on, the power-on password has to be entered to begin the boot process.

Each planar provides a minimum of 8 KB of non-volatile RAM (NVRAM). This memory is used to store configuration data for booting the system, and is split into different Open Firmware partitions. Each partition is either global or only available to Open Firmware. If a partition is global, it can be read from and written to by the firmware or the kernel.

4.2.2.3 Interrupt Controller

Each system board contains a chip that provides the functionality of dual 8259 interrupt controllers. These interrupt controllers work together in a master/slave configuration to serve the sixteen defined ISA interrupts. The interrupt controller, implemented on the I/O chip, handles the interrupts for the devices contained in the three PCI slots of the 43P. The F50 and S70 contain separate interrupt controllers that handle the interrupts for the PCI devices and buses.

The F50 uses an IBM MPIC interrupt controller that is compliant with the OpenPIC Register Interface Specification, Revision 1.0, and the S70 uses an OpenPIC interrupt controller that is compliant with Revision 1.2. Both the MPIC and the OpenPIC connect to the dual 8259 ISA interrupt controllers on their respective PCI-ISA bridge chips. The ISA interrupts are prioritized at the 8259 and delivered to the MPIC or OpenPIC for processing on one interrupt line.

All the registers contained within the MPIC and OpenPIC are memory mapped. The MPIC contains a shadow register for each processor in the system that is not software readable. This

register contains what interrupt is currently being serviced and its priority, for each processor. The shadow register is referenced by the MPIC when the MPIC is determining which processor should handle an interrupt. Each interrupt source handled by the MPIC has a collection of registers that maintain the priority of the interrupt and the possible destinations for the interrupt.

The OpenPIC is broken down into two separate functional units: the interrupt source unit (ISU) and the interrupt delivery unit (IDU). The S70 contains a maximum of sixteen PCI bridges, which requires a large number of interrupt lines. The ISU is integrated into the I/O chip that is a component of each I/O drawer. There are sixteen interrupt lines serviced per ISU. The IDU is a component of the RIO I/O chip, which is a component of the central electronics complex (CEC). The RIO I/O chip is directly connected to the SMP system data bus.

4.2.2.4 Serial and Parallel Ports

The serial and parallel ports are connected via the ISA buses in all three systems. The parallel I/O port is available on all three host computers for connection of an IBM Model 4317 printer. The serial ports are unused on all three computers in the evaluated configuration. The RS/6000 Trusted Facility Manual states that no devices should be connected via the serial ports.

4.3 Peripherals

4.3.1 Adapters

The SCSI adapters in the evaluated configuration are either located on the system main board or available as PCI plug-in cards. The 43P and F50 contain PCI SCSI-2 Fast/Wide adapters on board. The S70 has no SCSI adapters on board. All three computers can support PCI Single-Ended Ultra SCSI adapters installed in PCI bus slots.

There are four network adapters available for use in the evaluated configuration. The 43P contains on board hardware that is equivalent to the IBM PCI 10/100 MBPS Ethernet adapter. The F50 contains on board hardware that is equivalent to the IBM PCI 10Base5/T 10 MBPS Ethernet Adapter. There are no user addressable buffers or caches included on any of the network adapters in the evaluated configuration.

All the network adapters mentioned below operate as 32-bit DMA bus masters. The IBM PCI 10/100 MBPS Ethernet Adapter operates at either 10 MBPS (10BaseT) or 100 MBPS (100BaseT) full duplex by sensing the speed of the hub connection. It connects to Category-5 unshielded twisted pair (UTP) cabling via a RJ-45 connector. The Token-Ring PCI Adapter operates at either 4 MBPS or a 16 MBPS over a Token Ring LAN. The adapter connects to twisted pair cabling (shielded or unshielded) via an RJ-45 connector and automatically selects the correct LAN speed. The IBM PCI Auto LANstreamer Token Ring Adapter operates at either 4 or 16 MBPS, has an RJ-45 connector to twisted pair cabling.

There are no caches contained on the SCSI or network adapters, but there are buffers. These buffers are used in I/O accesses, and are implemented as small FIFOs. None of these buffers are addressable by any untrusted subject.

The GXT120P is a two-dimensional graphics adapter that is used with all three models in the evaluated configuration. The GXT120P provides a connection for the monitor, and contains two megabytes of SDRAM used as a frame buffer. There is a FIFO available, as well as the frame buffer, which is accessible to the user mode process that has control of the console. The graphics adapter provides no mechanism for storing commands for later use.

The rasterizer on the graphics adapter receives commands to perform drawing operations on the display. These commands allow solid and textured lines, short stroke vectors, polylines, bit, pattern and image bits, rectangle solid and pattern fills, and four point trapezoid solid and pattern fills. The rasterizer renders the drawing commands into pixels in the frame buffer. A RAMDAC chip translates the pixel values in the frame buffer into RGB for output to the display.

Appendix A contains a list of the hardware components that comprise the evaluated configuration.

4.3.2 Devices

The three systems provide various configurations of disk, CDROM, and tape devices, connected to one or more of the computer's PCI SCSI adapters. Each system must have at least one disk drive. Each system must also have a CDROM drive, in order to execute the CDROM-based installation.

The storage devices outlined below are not directly accessible by untrusted subjects. The information that these devices provide is only accessible to untrusted users at the system call interface. The kernel communicates with these devices using device drivers. Device drivers are discussed in section 5.3.4, I/O Management. Direct access to these devices is only performed during diagnostics, when the host is not in its secure state.

None of the SCSI devices used in the system have any user addressable caches or buffers.

Table 4-11. Devices. *The following devices are included in the evaluated configuration. The items marked Y for a column are usable on that system.*

Drive	43P	F50	S70
4.5 GB Ultra-SCSI	Y	Y	Y
20x ISO 9660 SCSI CD-ROM	Y	Y	Y
4mm SCSI tape drive	Y	Y	Y
9.1 GB Ultra-SCSI	Y	N	N
9.1 GB Ultra-SCSI hot-swappable disk	N	Y	Y
4.5 GB Fast/Wide SCSI DASD	N	Y	N
9.1 GB Ultra-SCSI DASD	N	N	Y

The 4MM tape drive contains firmware that may be updated. This firmware contains the microcode for the tape drive, and is updated using the **DIAG** command. The **DIAG** command sends updates over the SCSI bus using diagnostic tape commands. The firmware for the tape drive can also be updated using a firmware tape. The firmware update using a tape is not the normal method of upgrade provided by IBM outside of the evaluated configuration. The TFM instructs the administrator that it is possible to update the firmware using a firmware tape. The TFM also

states that only tapes that were written on the C2 system or received in original sealed package from the manufacturer should be used in the tape drive.

The SCSI hard disks contain firmware that may be updated. This firmware can only be updated from the system diagnostic routines. The firmware for the SCSI CD-ROM or floppy disk cannot be updated. The TFM states that no firmware may be updated, and that any modification of firmware violates the integrity of the TCB.

4.3.3 Service Processors

The F50 and S70 systems include service processors. A service processor is an embedded processor that is used in an SMP system to verify correct operation, and does not execute user mode processes. Neither service processor is accessible by untrusted subjects.

The only interface to modify the settings for the service processor requires that an ASCII terminal be connected to serial port one on the planar. The TFM states that no terminals should be connected to the serial ports in the evaluated configuration, so no modifications to the service processor configuration can take place.

The F50 service processor provides environmental regulation. If the internal temperature of the machine rises above a predefined limit, the service processor will increase the speed of the internal fans or shutdown the system.

The F50 service processor is physically located on the F50 planar. It is connected to the planar via the PCI bus and communicates with the CPUs in the system via interrupts.

The S70 service processor provides environmental regulation, as well as system initialization and an enhanced system integrity function. The S70 service processor contains the POST routines. These routines are executed at boot time. If the tests complete successfully, the service processor scans an initial state to the first CPU, and releases the CPU to continue the boot process.

The S70 service processor is notified on machine check exception conditions and for a set of correctable error conditions, which it analyzes and corrects them in real time. There is a threshold for correctable errors and if it is exceeded, the service processor will allow the machine check condition to stand which will effectively take the machine down. If a machine check exception occurs that can be recovered from, the service processor will stop the CPU, scan out data to analyze a problem and scan back in corrected data. The currently executing thread does not realize that its execution was interrupted.

The S70 service processor is physically located in the first I/O drawer. The S70 SP card contains the Super I/O chip that handles the ISA devices. The main CPUs in the system have control of the ISA bus. The service processor board provides the silicon space for the chip. The S70 service processor is connected to the I/O drawer using the PCI bus, and communicates with the CPUs in the system via interrupts.

4.3.4 Miscellaneous

The floppy drive, keyboard and mouse are connected using the ISA bus. These devices are connected to the dual-8259 interrupt controllers that are located in each system. The keyboard includes a buffer used to transfer keystrokes to the keyboard interrupt handler.

The monitor connects to the GXT120P video adapter, and receives data from the adapter. The monitor does not transmit any data to the adapter, nor does it offer any interfaces to facilitate such a transmission.

The keyboard, mouse, and monitor are exclusive use devices. Only a single user at a time may use these devices.

The operator panel display contains information about the current status of the machine. The operator panel display on each machine is implemented as an LED display, with three characters. The three characters are used during the boot process to report current status, and are used to report error messages in the event of a hardware failure.

The printer is an IBM Model 4317 Printer, attached to the host's parallel port. This printer supports Ethernet and Token Ring interfaces, but those options are prohibited by the RS/6000 Trusted Facility Manual. The printer is a multi-user device, but only services one print job at a time. The kernel appends a control sequence before and after each print job to clear the contents of the printers DRAM.

Ethernet hubs and Token Ring MAUs provide the central connection point for network cables from each host in the distributed system. These devices provide the electrical circuitry that allows the higher-level network protocols to communicate.

Any unintelligent Ethernet hub that is similar to the IBM 8222-008 or IBM 8222-016 or unintelligent Token Ring MAU that is similar to the IBM 8228 is acceptable in the evaluated configuration. None of these devices provide switching or routing of network packets, and none of these devices provide any interfaces.

4.4 Hardware Components Role in the Security Policy

Table 4-12. Device Summary. *This table summarizes the I/O buses, controllers, and devices in the ToE with respect to involvement in the security policy.*

Device	User-Accessible	User-Programmable	Residual Data
Level 1 cache	Yes, but not addressable	No	Yes
Level 2 cache	Yes, but not addressable	No	Yes
PowerPC bus logic	No	No	No
PCI bus logic, bus bridge	No	No	No
ISA bus logic	No	No	No
S70 RIO Bus logic	No	No	No
PCI SCSI-2 Adapter	No	No	Yes
PCI SCSI Adapter	No	No	Yes

Device	User-Accessible	User-Programmable	Residual Data
SCSI Hard Disks	Yes, via file system only	No	Yes
SCSI CDROM	Yes, but only via file system. Install script changes rights to device file to prevent access.	No	No
4mm SCSI Tape	No, the install script changes rights on device file so access is limited to root.	No	Yes
Diskette	No, as follows: TFM guidance not to mount file system from floppy. The install script changes rights on device file so access is limited to root.	No	Yes
GXT120P 2-D Graphics Adapter	Yes, frame buffer mapped directly into user process	Yes, the RCM maps in whatever registers (or more generally command FIFOs) are required for rendering.	No
Monitor	Yes	Not programmable at all, except for features such as brightness, contrast, and screen size.	No, the login prompt clears the screen of data from the previous session. It outputs a large number of new lines before showing the login prompt.
LAN Adapters	Yes. Users may send TCP/UDP messages, but cannot send or read raw packets, or create raw sockets.	No	Yes, but data is not accessible by subsequent users.
Keyboard	Yes	No	No
Mouse	Yes	No	No
Operator Panel Display	Yes	No	Minimal
Service Processors (F50 and S70)	The Service Processor requires the privilege access password to view or make changes, and only provides an ASCII terminal interface through serial port one. The TFM forbids the use of an ASCII terminal to access service processor configuration.	No	Yes, but does not provide an interface for administrative or non-administrative users.
IBM Model 4317 Network Printer	Yes. A user may submit a print job for a specific printer.	Yes	No. The kernel sends a control sequence at the beginning and end of each job to clear the printers buffer.
Serial ports	No (TFM warning not to connect anything to serial ports)	No	No
Parallel port	Yes	No	No
Firmware	No	No	No
Power Management HW	Indirectly, by stopping input and letting the host enter sleep mode.	No	No

4.5 Hardware Input/Output

Hardware I/O is patterned after the same architecture on all three machines, and is performed through memory mapped access to adapters and associated registers. There are no user mode I/O instructions on the 604e or RS64. Areas used in hardware I/O are under the control of the TCB until an interrupt signals completion, and the kernel allows a thread to continue execution.

When Open Firmware begins to initialize the system, it probes the various PCI buses in the system and creates a device tree with all the relevant information concerning memory addresses in use by the various adapters. This device tree is transferred to AIX during the boot process, and defines which device drivers can be loaded.

Each system contains at least one PCI bus for connecting adapters. A PCI host bridge is used to connect the PCI bus to the memory I/O controller. The PCI Host Bridge determines the bus memory space, the bus I/O space, and the PCI configuration space. The PCI configuration space is populated by Open Firmware, using the PCI slot and direct lines on the system board.

Programmed I/O is used for slower ISA devices, such as the keyboard and mouse. When one of these devices has data to send, it generates an interrupt. During the execution of the interrupt handler, the single character is received and acted on accordingly.

Device drivers in the kernel perform DMA. DMA reads and writes to memory addresses are performed using real addresses. The real addresses are mapped into the real address space during system initialization.

There are two different types of I/O devices in the evaluated configuration: those with fixed address assignments and those with programmable base registers. The fixed address devices are basic devices needed for the overall operation of the system. An example of a fixed address device is the interrupt controller. The memory addresses of fixed devices are hard-wired into the system. The programmable devices are configured with a base address that determines the starting address for the adapter's memory range.

During system initialization, each PCI adapter card is probed to determine how much of its adapter memory is addressable. This memory includes any memory-mapped registers or buffers. Each adapter has a base address register that determines the base memory address for the adapter. The base address is the address in the real address space where this adapter's range begins. The kernel's I/O subsystem communicates with the adapter using a real memory address in this range. This portion of the address space is not managed by the VMM, because there are no physical pages of memory associated with the memory addresses.

DMA operations can be performed by the kernel internally or directly to a page in the users address space. Any process that is receiving the results of a DMA operation in a page of their address space must have that page hidden and pinned by the kernel. This prevents the page from being swapped out or viewed when the kernel is copying data, possibly leaving the memory location in an inconsistent state.

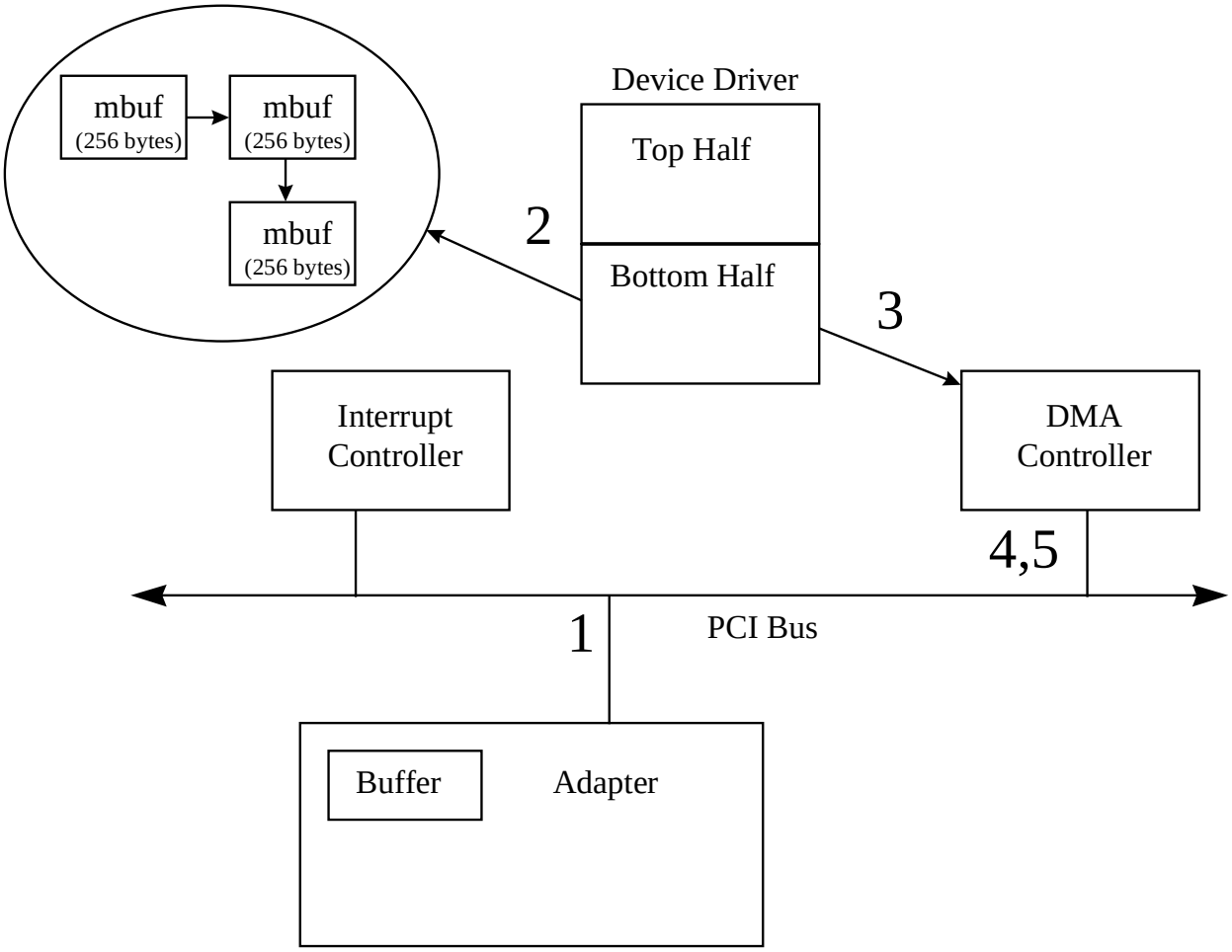

Figure 4.2: DMA Operation for an Ethernet Receive

1. The Ethernet adapter generates an interrupt when it receives a packet from the wire.
2. The top-half of the device driver maintains a pool of kernel memory buffers (mbuf). When the interrupt is received, the top-half of the device driver allocates memory buffers for this transfer and pins them in memory. The device driver itself knows the number of memory buffers to allocate. In this example, the Ethernet adapter has an MTU size of 1500, so a linked list of mbufs are allocated to contain 1500 bytes.
3. The bottom-half of the device driver passes the DMA controller a pointer to the mbuf chain and configures the DMA controller registers to pick up the data contained on the Ethernet adapter buffer.
4. The DMA controller performs the DMA operation, moving the contents of the Ethernet buffer to the mbufs.
5. The DMA controller generates an interrupt when the operation is complete. The bottom-half of the device driver releases the mbufs to the device driver's pool.

When the kernel has a packet to send, the top-half of the device driver sets up the DMA controller for the operation, and configures the adapters memory mapped registers to prepare the adapter to receive the packet.

The only difference in the above scenario for different types of adapters is the size of the buffers and the location of the buffers within the kernel address space. Different adapters use buffers from other sources, such as pages managed by the VMM being used for disk I/O.

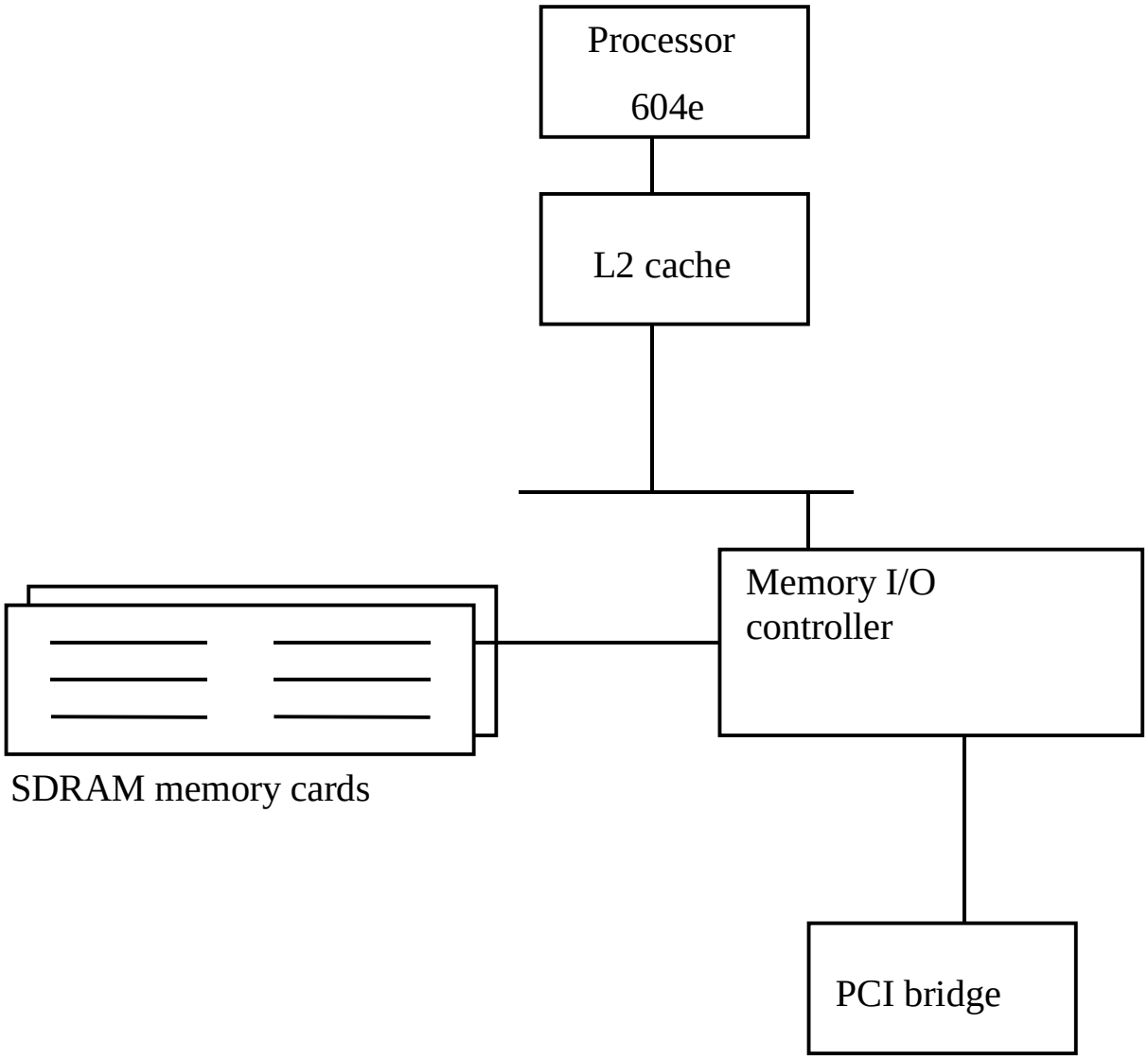

Figure 4.3: 43P Overview Diagram

The 43P provides one single PCI bus, connected to the Memory I/O controller.

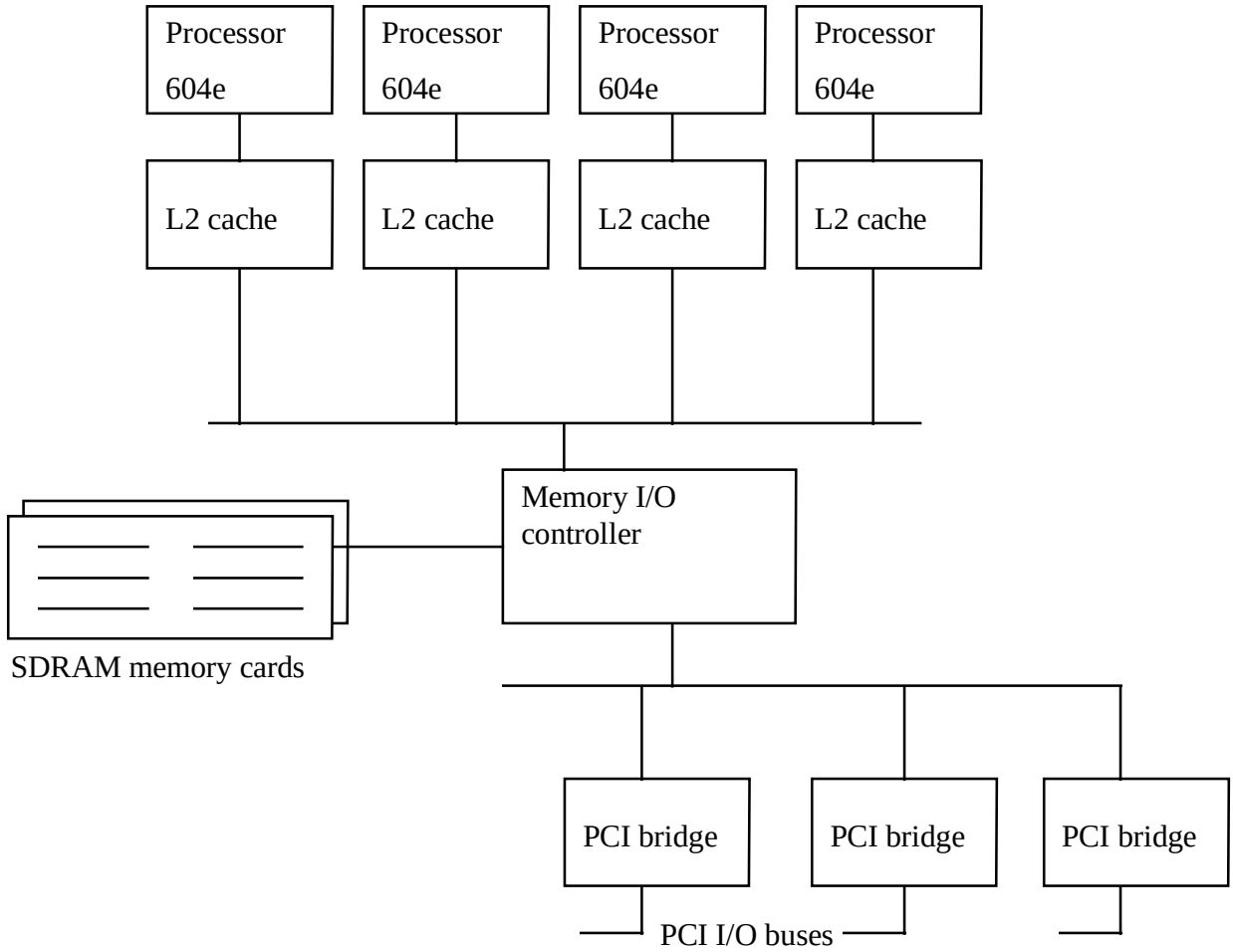

Figure 4.4: F50 Overview Diagram

The PowerPC Local Bus on the F50 connects the four processors and L2 caches. The I/O bridge bus connects the PCI I/O bridges with the Memory I/O controller unit. The PCI Bridge chips each support one PCI bus.

The S70, see figure 4.5, is implemented as separate racks for the processors (CEC) and the I/O drawers. Each I/O drawer has four PCI bridges, for a total of fourteen PCI slots per drawer. One of the PCI bridge chips drives two 64-bit PCI slots, while the other three each drive one 64-bit PCI slot and three 32-bit PCI slots. The PCI bridges are connected into the I/O Bridge, which connects to the I/O Hub in the system rack. The I/O Bridge acts as a converter from PCI to RIO, and the I/O hub on the system rack side converts from RIO back to PCI.

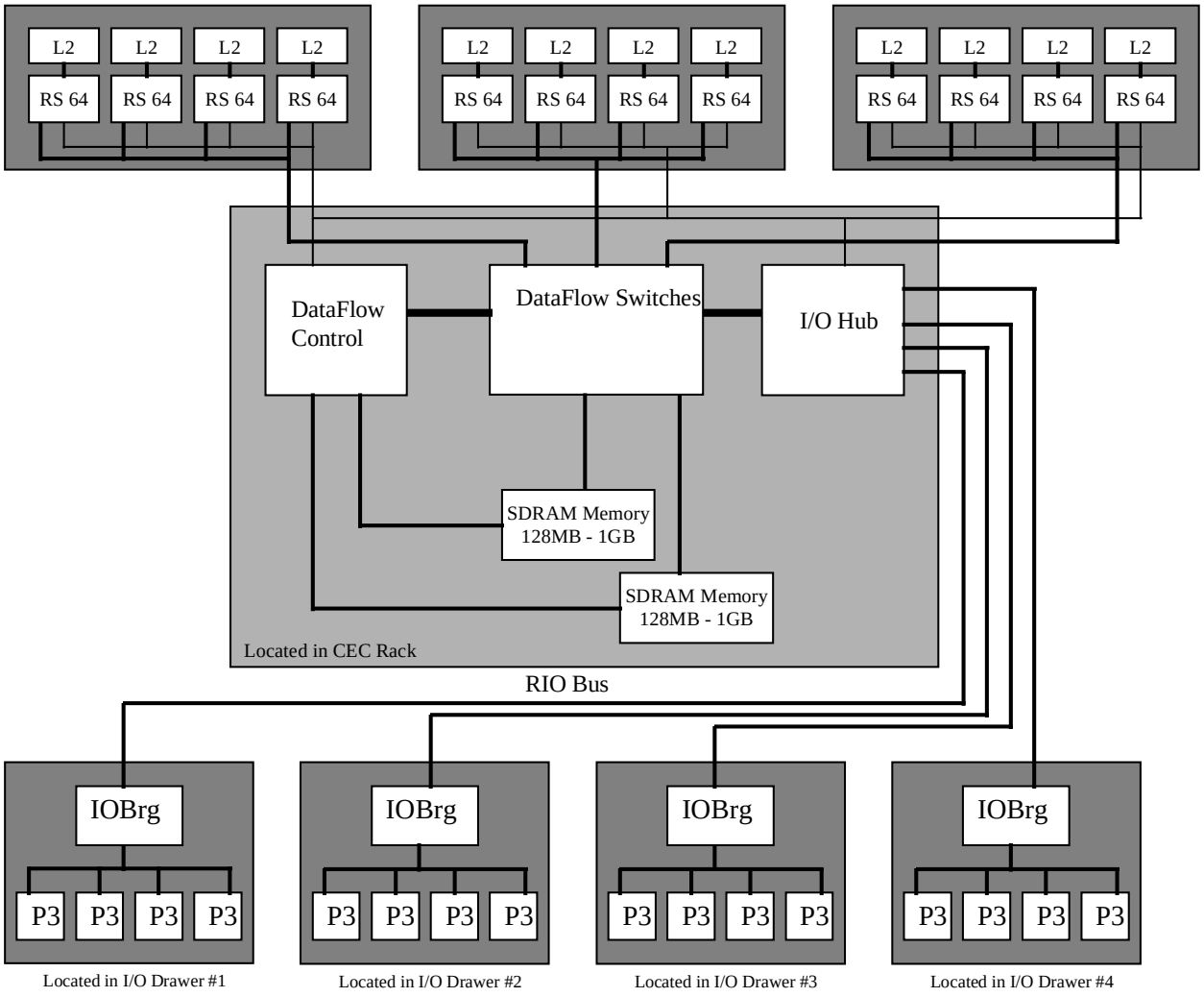

Figure 4.5: S70 Overview Diagram

4.5 Multiprocessing

Two models included in the evaluation contain multiple processors: the F50 and S70. The F50 is a four-way SMP machine, using the 604e processor and the S70 is a twelve-way SMP machine, using the RS-64 processor. The S70 processors are contained on three separate CPU cards, each containing four processors.

The hardware definition of SMP is that all processors have equal access to system memory. The F50 and S70 implement the sharing of system memory through the use of the PowerPC Local Bus. Each CPU in both systems connects to this bus.

Cache coherence is maintained using the MESI four-state cache coherency protocol. This cache protocol is referred to as four-state because each cache block in the data cache is in one of the four states. The instruction caches do not implement the MESI cache coherency protocol, but instead uses a single state bit to control whether the instruction block is valid or not.

The MESI cache coherency uses a broadcast technique to maintain a coherent memory system. Each processor must broadcast it's intention to read a cache block not in the cache, and must also broadcast it's intention to write into a block that is not owned exclusively. Other processors or devices respond by searching their own caches and reporting back status to the originating processor. The response will either be a shared indicator or a retry indicator. A shared indicator specifies that the processor is also using that particular cache block. A retry indicator specifies that the processor has either a copy of the cache block it needs to return to system memory or a queuing problem occurred that prevented the processor from doing the proper snooping operation.

Table 4-13. MESI Cache Coherency Protocol. *MESI is a technique to maintain a coherent memory system on a multiprocessor system.*

MESI State	Definition
Modified (M)	This block of memory is only valid in this cache. The block has been modified from system memory, and has not yet been written back.
Exclusive (E)	This block of memory is in this cache only. The data is consistent with system memory.
Shared (S)	This block of memory is valid in the cache and in at least one other cache. The shared state is shared-unmodified. There is no shared-modified.
Invalid (I)	This block of memory is not resident in the cache and/or any data contained is considered not useful.

The interrupt controllers in the F50 and S70 distribute interrupts to the collection of CPUs. The specifics about each interrupt controller are contained in section 4.2.2.3, Interrupt Controller.

When an interrupt is generated in the multiprocessor environment, the interrupt controller checks the priorities of all the processors. The interrupt is routed to the processor that has a priority level allowing it to service the interrupt. Interrupts can be nested if a processor is servicing an interrupt, and an interrupt of higher priority is routed to it. The current lower priority interrupt will be saved and the higher priority interrupt will be processed.

4.6 Memory Architecture

The memory management tasks of each host in the system are split between the hardware and software. The hardware translates effective addresses to physical addresses and protects pages, segments, and blocks of memory. The software aspects of memory management are discussed in section 5.3.1, Memory Management.

4.6.1 Segmentation

The memory management architecture of the two processors is based on a segmented memory model. A segment is a 256MB piece of virtual memory containing 65,536 pages, each 4096 bytes, and configured through a segment descriptor. Pages are described in section 4.6.3, Paging.

4.6.1.1 32-Bit Segmentation

There are sixteen segment registers in the PowerPC 604e processor. The segment registers define a collection of properties about the segment, including the virtual memory segments that make up the process address space.

The kernel can modify the segment registers, using the move to segment register and move from segment register instructions. User mode software is unable to modify the segment registers because the move to segment register (mtsr) and move from segment register (mfsr) instructions are supervisor mode instructions. This prevents a user mode process from directly changing which segments make up the process' address space.

Table 4-14. Segment Register Format. *Each segment register contains a format bit, a collection of protection bits, and a Virtual Segment ID.*

T	Ks	Kp	N	Reserved	VSID
0	1	2	3	4-7	8-31

Table 4-15. Segment Register Fields. *The segment register contains the following fields.*

Field	Description
T	Selects the format of the segment descriptor (0 = page address translation)
Ks	Supervisor state protection key
Kp	User-state protection key
N	No-execute protection bit
VSID	Virtual Segment ID

4.6.1.2 64-Bit Segmentation

The RS-64 processor can be operating in either 32-bit or 64-bit mode. The AIX kernel always runs in 32-bit mode. 64-bit mode is only used for 64-bit processes. The MSR contains a bit that determines the current mode: 32 or 64 bit.

4.6.1.2.1 64-Bit Mode

The segments that make up a 64-bit process executing on the RS64 processor are cached in the segment table and a Segment Look-aside Buffer (SLB), and are directly referenced through the process context. The RS-64 processor provides two separate SLBs: a four entry instruction SLB and an eight entry data SLB. The SLB is an on-chip cache of the segment table, and is used during the effective to virtual address translation process.

64-bit processes use an array to keep track of what segments the process is using. The array of segments is stored in the process private segment. This area of the process private segment is only available to the kernel. One instance of the segment table exists for each 64-bit process. The segments that make up a 32-bit or 64-bit process are discussed in section 5.3.2.1, Typical Process Address Space.

Table 4-16. Segment Table Entry and SLB Format. *The segment table provides a cache of the 256 most recently used effective segment ids to virtual segment ids. The SLB caches a subset of the current segment table on-chip.*

ESID	Reserved	V	T	Ks	Kp	N	VSID	Reserved
0-35	36-55	56	57	58	59	60	0-51	52-63

Table 4-17. Segment Table Entry and SLB Format. *Each segment table entry contains an effective segment id, a format bit, a collection of protection bits, and a Virtual Segment ID.*

Field	Description
ESID	Effective Segment ID
V	Entry valid if V=1
T	Selects the format of the segment descriptor (0 = page address translation)
Ks	Supervisor state storage key
Kp	Problem state storage key
N	No-execute protection bit
VSID	Virtual Segment ID

4.6.1.2.2 32-Bit Mode

The address translation unit hardware provides a level of abstraction for memory management operating in 32-bit mode on the 64-bit processor. When the kernel sets up a 32-bit process' address space on the 64-bit processor, it uses the same instructions to move to and from segment registers as are used on the 32-bit 604e.

The 64-bit processor uses a 32-bit Bridge to emulate the segment registers. The 32-bit bridge is transparent to the kernel and is included to facilitate PowerPC operating system vendors who are transitioning from 32-bit to 64-bit kernels. By using the abstraction provided by the 32-bit Bridge, IBM did not have to move to a 64-bit kernel to support the RS-64.

When the 64-bit CPU is operating in 32-bit mode, the SLB is used to represent the sixteen segment registers that make up the process address space. These segment registers are referred to as virtual, because they do not store the same exact values as the 32-bit segment registers. The virtual segment register mappings contained within the SLB use the same structure as 64-bit mode.

4.6.2 Address Translation

There are three types of addresses used in memory management: real, effective and virtual. Real addresses point to the actual, physical memory location being addressed; effective addresses are 32-bit or 64-bit addresses specified for load, store, or instruction fetches. Virtual addresses are temporary addresses used to translate effective addresses into real addresses.

Effective addresses are not unique. Each process begins execution of a program at the same effective address. When a process requests an instruction using an effective address, a virtual address is used to translate to a real memory address. This virtual address contains the virtual segment ID, which is the segment that contains this effective address. Virtual addresses and virtual segment IDs are unique for user mode process segments that are not shared (process

private segment). For a discussion of the processes address space and how some segments are shared, see section 5.3.2.1, Typical Process Address Space.

The Translation Look-aside Buffer (TLB) used with each processor is a cache for the most recently used page table entries. The 32-bit and 64-bit implementations of the TLB use the same format as the page table entries. The 604e and RS64 each have separate TLBs for instructions and data. The 604e TLBs hold 128 entries each, while the RS64 TLBs hold 512 entries each.

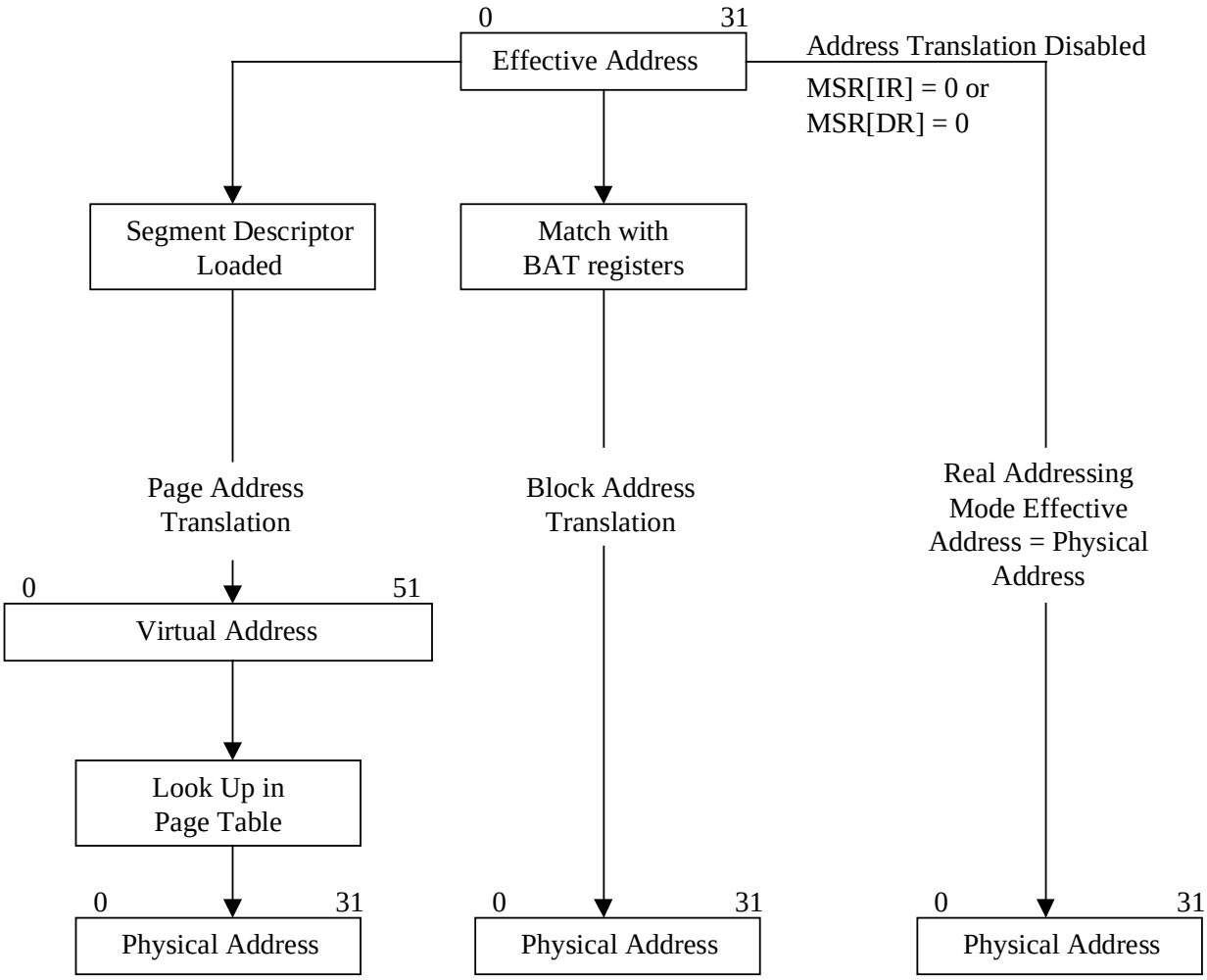

Figure 4.6: Address Translation - General Overview

4.6.2.1 32-bit Address Translation

32-bit address translation is the process of translating 32-bit effective addresses to 32-bit real addresses. To make this translation, 52-bit virtual addresses must be derived.

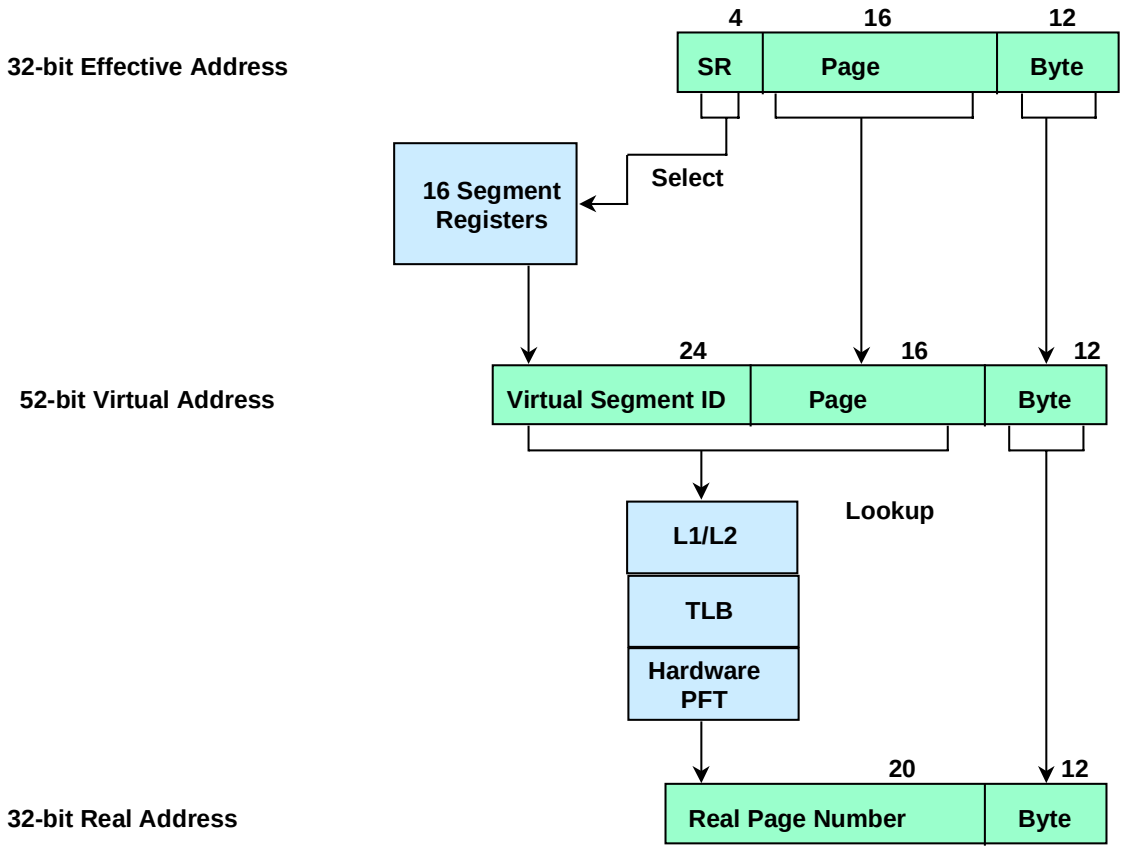

Figure 4.7: Address translation overview (32-bit implementations)

The PowerPC 604e processor's effective address is made up of three parts: the segment register number, the page number within the segment, and the byte offset within the page. The segment register number points to a segment register containing the Virtual Segment ID (VSID). Each segment register defines a 256MB piece of the process' virtual address space. The VSID and a hashed page index are referenced in the L1/L2 cache, the TLB and the hardware page frame table searching for a match. The byte offset within the page is carried from the effective address and inserted into the real address. If a match is not found in the L1/L2 cache, the TLB, or the hardware page frame table, a data storage interrupt is generated and the VMM takes over.

32-bit address translation on the 64-bit hardware is identical in function to the 32-bit hardware. Figure 4.7 is accurate for 32-bit address translation on both the 32-bit and 64-bit hardware. The 64-bit CPU provides additional hardware and structures that are used with the 32-bit CPU, but this additional hardware is not visible to the kernel.

4.6.2.2 64-bit Address Translation

64-bit address translation is the process of translating 64-bit effective addresses into 64-bit real addresses. To make this translation, 80-bit virtual addresses must be derived.

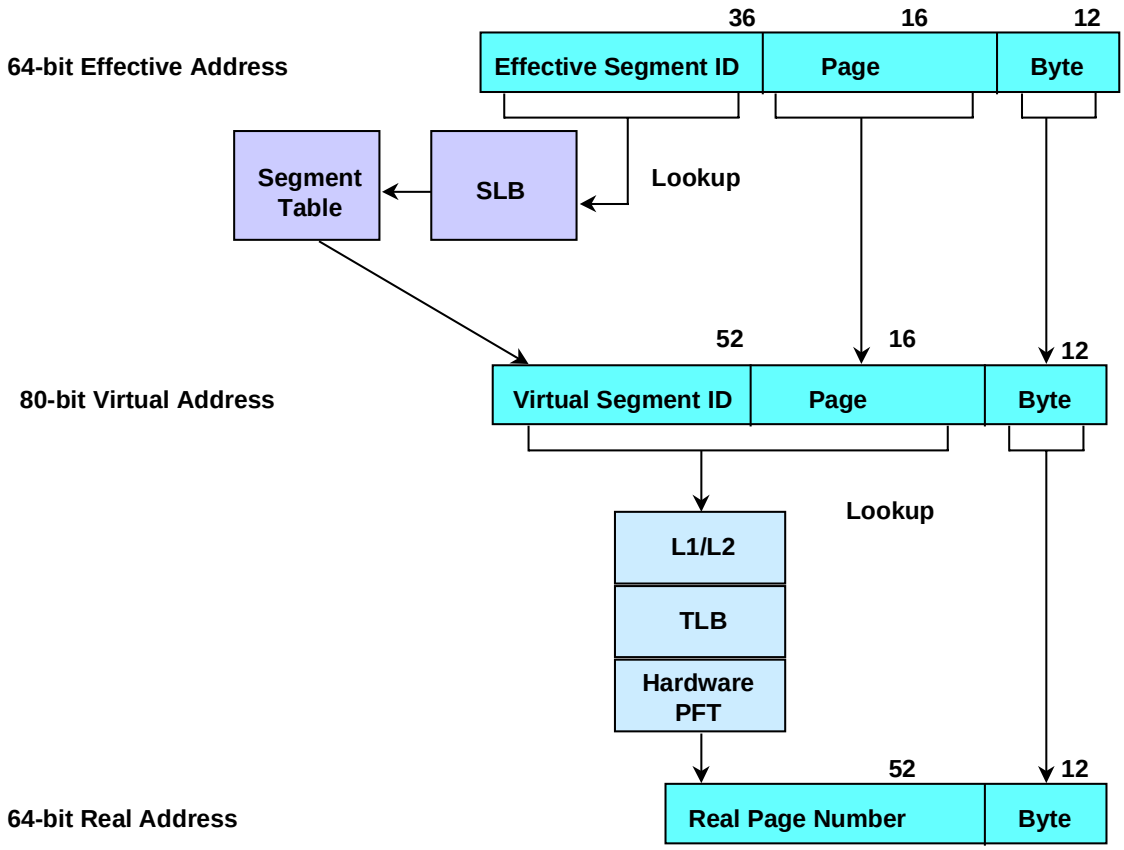

Figure 4.8: Address translation overview (64-bit implementations)

The PowerPC RS64 processor's effective address is made up of three parts: the effective segment ID, the page number within the segment, and the byte offset within the page. The effective segment ID is looked up in the on-chip SLB and Segment Table to produce a Virtual Segment ID (VSID). If the ESID is not found in the Segment Table or the on-chip SLB, a segment fault occurs. The VMM searches the process context structure containing the effective segment ID to VSID mappings and updates the segment table and SLB if a match is found. The VSID and a hashed page index are referenced in the L1/L2 cache, the TLB and the hardware page frame table searching for a match. The byte offset within the page is carried from the effective address and inserted into the real address. If a match is not found in the L1/L2 cache, the TLB, or the hardware page frame table, a data storage interrupt is generated and the VMM takes over.

4.6.2.3 Block Address Translation

Block address translation (BAT) maps ranges of effective addresses that are not subject to paging into contiguous areas of real memory. Block address translation uses a different breakdown of the bits of the effective address. The last seventeen bits of the effective address are used to specify the offset within the block, while the first fifteen bits define the location of the block in the effective address space. The first fifteen bits are translated to the real address of the block, which is concatenated with the 17-bit offset. See figure 4.9.

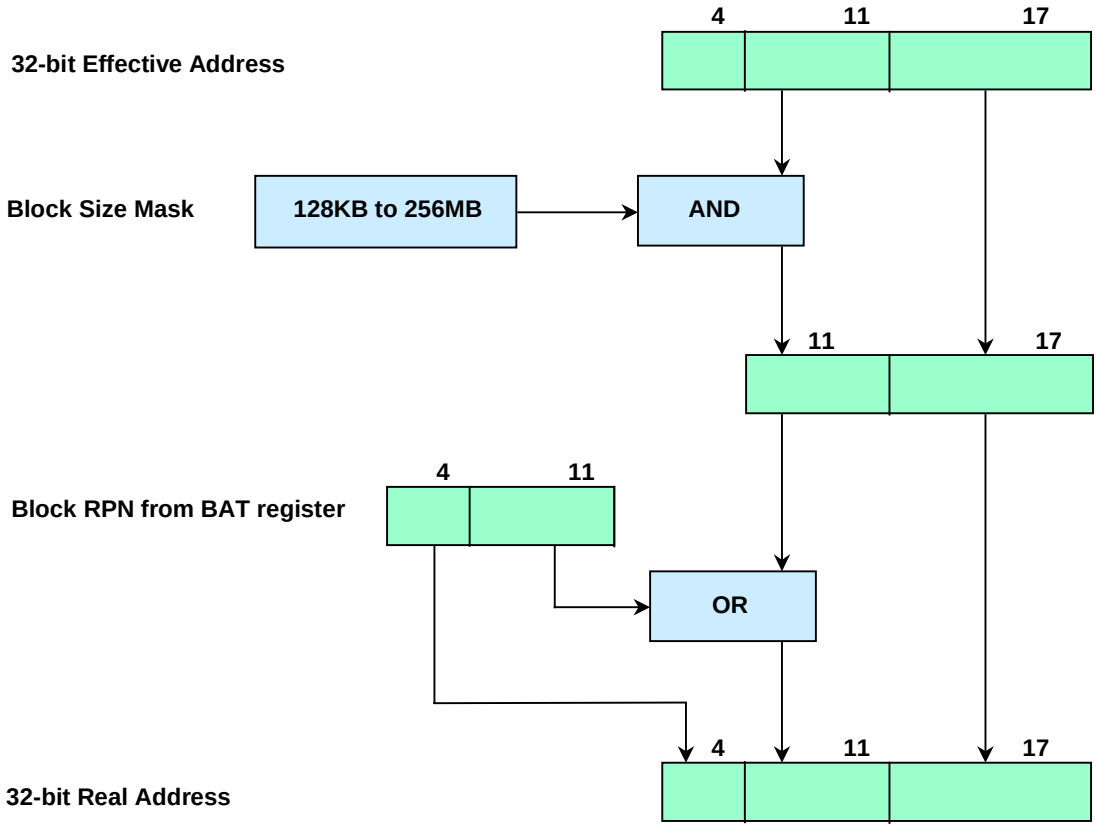

Figure 4.9: Block Address translation (32-bit implementations)

Block address translation is controlled by BAT register pairs. The BAT registers are supervisor state registers, so they are not modifiable by untrusted users. The kernel and device drivers use BAT to translate their address space to physical memory. If the BAT registers are mapped for an effective address, the translation will succeed if either the MSR indicates that the CPU is operating in Supervisor State or the protection bits are set to allow user mode access to the block.

Table 4-18. Upper BAT Register Format. *The Upper BAT register contains the effective page index, the BAT length, and the supervisor and problem state valid bits.*

BEPI	Reserved	BL	Vs	Vp
0-14	15-18	19-29	30	31

Table 4-19. Lower BAT Register Format. *The lower BAT register contains a field used with BL from the upper BAT register to generate the high order bits of the physical address of the block, the cache control bits, and the protection bits for this block.*

BRPN	Reserved	WIMG	Reserved	PP
0-14	15-24	25-28	29	30-31

The 4-bit segment register designation is ignored for block address translation. The block size mask is subtracted from bits 4-14 to strip away the high order bits of the effective address. The result is added with the high order bits stored in the block real page number. The first 4 bits of the block RPN provide additional high order bits. The 17-bit offset within the block is carried from the effective address to the real address.

This mechanism is available to user-mode processes, provided the kernel sets up the BAT registers. The video adapter driver uses BAT register pairs to map video data into a user's process address space.

4.6.3 Paging

Table 4-20. Page Table Entry Format, stored in the Hardware Page Frame Table, Word 0 and Word 1 (32-bit). *The page table tracks the pages that are in use and a number of attributes associated with each page.*

V	VSID	H	API
0	1-24	25	26-31

RPN	Reserved	R	C	WIMG	Reserved	PP
0-19	20-22	23	24	25-28	29	30-31

Table 4-21. Page Table Entry Format, stored in the Hardware Page Frame Table, Word 0 and Word 1 (64-bit). *The page table tracks the pages that are in use and a number of attributes associated with each page.*

VSID	API	Reserved	H	V
0-51	52-56	57-61	62	63

RPN	Reserved	R	C	WIMG	Reserved	PP
0-51	52-54	55	56	57-60	61	62-63

Table 4-22. Page Table Entry Fields (32-bit and 64-bit). *The page table tracks the pages that are in use and a number of attributes associated with each page*.

Field	Description
V	Entry valid if V=1
VSID	Virtual Segment ID
H	Hash function identifier
API	Abbreviated page index
RPN	Physical page number
R	Referenced bit
C	Changed bit
WIMG	Memory/cache control bits
W	Write-Through Attribute
I	Caching-Inhibited Attribute
M	Memory Coherency Attribute
G	Guarded Attribute
PP	Page protection bits

Paging is the mechanism of swapping small pieces of memory in and out of real memory to create a larger virtual address space. A page is a 4096 byte unit of memory. When a piece of memory is swapped out, it is placed on paging space. Paging space is a disk area used by the memory manager to hold inactive memory pages. Both hardware and software implement paging.

4.6.4 Memory Protection

The memory protection mechanisms used in the system provide similar protection schemes for segments, pages and blocks. The memory management hardware protects supervisor areas from untrusted user access and designates areas of memory as read-only or read-write.

4.6.4.1 Segment Protection

Only privileged software in the kernel can load the segment registers that define the process address space. A process cannot attempt to access memory that is not in one of its segments, because the segment registers are used to determine the real address being referenced, and only the kernel can modify the values in the segment registers.

The Supervisor State protection key (Ks) or the Problem State protection key (Kp) is combined with the protection bits (PP) to determine page level access. Section 4.6.4.2, Page Protection explains how protection keys and protection bits combined determine the page access level. The No-execute protection bit (N) is architecturally defined and present in both processors, but is not used by AIX.

4.6.4.2 Page Protection

The page memory protection mechanism provides access to a page in memory for supervisor state or problem state software. The PP bits are stored in the Page Table Entry (PTE) for a particular page. Each page access check references the current state of the machine as defined in the MSR, the supervisor or problem state protection keys and the page protection bits to determine if the access is allowed. Page memory protection cannot be used to provide access for individual users.

Memory access checks are performed by the MMU in hardware. Following the successful translation of a memory address, the MMU checks the protection key and the page protection bits. If the access is disallowed, an exception is generated by the hardware and the access is not granted. The kernel does not perform the access check for memory.

Table 4-23. Protection Key and Page Protection.

Ks or Kp	Page Protection Bits	Page Access
0	00	Read/Write
0	01	Read/Write
0	10	Read/Write
0	11	Read Only
1	00	No Access
1	01	Read Only
1	10	Read/Write
1	11	Read Only

4.6.4.3 Block Protection

The BAT registers for the 32-bit and 64-bit processors each contain a Supervisor mode bit, a User mode bit, and two protection bits. When either the supervisor or user bits are enabled, the

two protection bits determine access rights to the memory address block. The result is access to the specific block or an exception.

4.6.4.4 Protection Violation

A protection violation occurs when a user or supervisor attempts to read or write to a memory address that they do not have access to. When a protection violation occurs, an ISI exception is generated if the address was being translated for an instruction access. A DSI exception is generated if the address was being translated for a data access.

4.7 Context Switching

4.7.1 Registers and Threads

Each thread that runs on the system saves the contents of its hardware registers when a context switch occurs. The registers are stored in a structure that is part of the thread context, accessible from the process table entry for that process and managed by the kernel. A context switch may also occur when an interrupt saves the state of the machine and executes its handler. Interrupts are described in section 4.1.3, Interrupts and Exceptions.

When a context switch occurs, the resident values of the CPU registers are overwritten by the values of the incoming thread, negating the chance of any residual data being saved in a CPU register.

The following registers are stored within the context of a 32-bit thread: instruction address, machine state, condition, link, count, floating-point status, segment registers, general purpose registers, and floating point registers. A 64-bit process does not save segment registers, since they do not exist in this context, and adds the address space register.

4.7.2 Floating Point and Threads

The status of the floating-point registers is not saved each time a context switch occurs. One thread at a time has ownership of the floating point registers on a processor. A field contained within the per-processor data area (PPDA) determines the ownership. The PPDA is a per-processor kernel structure.

A bit in the MSR determines whether the current thread can execute floating-point instructions. If a thread does not have this bit enabled, and attempts to execute a floating-point instruction, a floating-point unavailable exception is generated.

On a single processor system the floating-point unavailable handler saves the values for the floating-point registers to the owning thread, assigns ownership of the FP registers to the current thread, and returns to retry the instruction. On a multiprocessor system, the FP register values are always saved. The floating-point unavailable handler enables the bit to allow floating-point instructions to be executed and returns to retry the instruction.

4.7.3 Cache

The L1, L2 and TLB caches are not cleared when a context switch occurs, so the L1 and L2 caches may contain cache-lines from a previous process, and the TLB may contain page table entries from a previous process. The currently executing user process cannot generate a virtual address that points outside of their process address space. The currently running process overwrites the values stored in the cache from the previous process.

4.8 Hardware Equivalency

There are no security relevant differences between the PowerPC 604e and PowerPC RS64 processors. The domain separation model is conceptually the same. There are slight differences between the processors with respect to segmentation, instruction sets, and supported peripherals, but there are no significant differences between the two architectures from a security relevance perspective. These two processors are functionally equivalent.

The SCSI adapters all provide a connection between the I/O hardware on the system board and SCSI devices. The implementation of Ultra-SCSI versus Fast/Wide-SCSI provides no security relevance. Ultra-SCSI contributes faster transfer rates than Fast/Wide. The absence of any untrusted user addressable buffers or caches further abstracts the individual SCSI adapters as devices that provide I/O. The SCSI devices differ in the amount of disk space they provide, and their implementation of a particular SCSI standard (Ultra versus Fast/Wide).

The network adapters provide an interface for the kernel to the network cabling. The distinction between token ring and Ethernet is quite different, but not security relevant. Token ring uses a token system to determine which network adapter has the ability to send packets in a given time frame, and Ethernet uses a broadcast mechanism. These differences are not security relevant, as they just define the mechanism for packets to be transferred. The use of the promiscuous mode of Ethernet is controlled by the kernel on each individual host, blocking any access an untrusted user may have to broadcast network traffic. None of the network adapters contain any untrusted user addressable buffers or caches.

5. TCB SOFTWARE

This chapter presents the general structure of the RS/6000 Distributed System Trusted Computing Base (TCB) software, identifies TCB software components, and summarizes the operation of each. The descriptions in this chapter concentrate on functional properties. Although protected resources and security policies are mentioned here, complete descriptions are in the two following chapters. This chapter begins with a general characterization of TCB software classes then describes each major TCB software component.

5.1 TCB Structure

This section identifies the major classes of software components, and describes the structuring mechanisms by which they are organized. Because isolation and protection are critical to the reference monitor concept, those aspects of the TCB are discussed first, followed by actual software descriptions. Although this introduces numerous concepts not explained until later in the chapter, it serves as a bridge to the descriptions that will follow.

5.2 TCB Definition

The AIX TCB software is defined as all the software that either is responsible for implementing the security policy, or that could affect correct operation of the security policy.

TCB components are further classed as significant or non-significant, depending on whether they have direct responsibility for implementing or enforcing the security policy. Because the AIX TCB is in no sense minimal, the TCB contains many components that are present for functional reasons or programming convenience, and are not strictly necessary for implementation of the security policies. Those components (for example: utilities, libraries, I/O device drivers) are the non-significant ones, and are mentioned here only when they are crucial to understanding the system's architecture. They are trusted to function correctly to the extent that they do not contain malicious code and do not affect the correct operation of the security mechanisms.

5.2.1 TCB Isolation Argument

This presents the TCB isolation or reference monitor argument, which is recapitulated in the System Architecture requirement section in Chapter 9. This argument shows that, to the degree appropriate for the TCSEC C2 level of trust, the RS/6000 Distributed TCB is tamper-resistant and always invoked.

The RS/6000 Distributed System TCB is tamper-resistant because all TCB programs, data, and other components are protected from unauthorized access via numerous mechanisms. The security policies are described further in Chapter 7, and architectural aspects of TCB protection are described later in this chapter.

To an extent, the kernel is functionally organized. There are separate source code files or groupings of source code that contain the programs that implement specific kernel functions (e.g., file system, process management). However, this separation is a design abstraction, and data structures are not strictly isolated.

All kernel software has access to all areas of memory, and the ability to execute all instructions. In general, however, only memory containing kernel data structures is manipulated by kernel software. Parameters are copied to and from process storage (i.e. that accessible outside the kernel) by internal mechanisms, and those interfaces only refer to storage belonging to the process that invoked the kernel (e.g., by a system call).

5.2.1.1 TCB Protection

While in operation, the kernel TCB software and data are protected by the hardware memory protection mechanisms described in Chapter 4, section 4.1.1, Execution States and section 4.6.4, Memory Protection. The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes.

Non-kernel TCB software and data are protected by DAC and process isolation mechanisms. In the evaluated configuration, the reserved user ID root, or other reserved IDs equivalent to root, owns TCB directories and files In general, TCB files and directories containing internal TCB data (e.g., audit files, batch job queues) are also protected from reading by DAC permissions.

The TCB hardware and firmware components are required to be physically protected from unauthorized access. The system kernel mediates all access to the hardware mechanisms themselves, other than program visible CPU instruction functions.

The boot image for each host in the distributed system is adequately protected. A description of the boot logical volume can be found in section 5.3.16, Initialization and Shutdown.

5.2.1.2 TCB Invocation Guarantees

All system protected resources are managed by the TCB. Because all TCB data structures are protected, these resources can be directly manipulated only by the TCB, through defined TCB interfaces. This satisfies the condition that the TCB must be "always invoked" to manipulate protected resources.

Resources managed by the kernel software can only be manipulated while running in kernel mode. Processes run in user mode, and execution in the kernel occurs only as the result of an exception or interrupt. The TCB hardware and the kernel software handling these events ensure that the kernel is entered only at pre-determined locations, and within pre-determined parameters. All kernel managed resources are protected such that only the appropriate kernel software manipulates them.

Trusted processes implement resources managed outside the kernel. The trusted processes and the data defining the resources are protected as described above depending on the type of interface. For directly invoked trusted processes the program invocation mechanism ensures that

the trusted process always starts in a protected environment at a predetermined point. Other trusted process interfaces are started during system initialization and use well defined protocol or file system mechanisms to receive requests.

5.2.2 Relationship to UNIX Systems

Because the RS/6000 Distributed System is derived jointly from System V and Berkeley versions of the UNIX system, it has a structure much like them: privileged kernel and processes, some trusted, some not. The primary novel aspects of the system are in the area of a dynamically extensible kernel, support for real-time processing, and enhancements to the traditional system management paradigm. These are discussed throughout the report as they become relevant.

5.2.3 Kernel

The RS/6000 Distributed System software consists of a privileged kernel and a variety of non-kernel components (trusted processes). The kernel operates on behalf of all processes (subjects). It runs in the CPU's privileged mode and has access to all system memory. All kernel software, including kernel extensions and kernel processes, is part of the TCB. The kernel is entered by some event that causes a context switch such as a system call, I/O interrupt, or arithmetic error. Upon entry the kernel determines the function to be performed, performs it, and, when finished, performs another context switch to return to user processing (possibly on behalf of a different subject).

The kernel is shared by all processes, and manages system wide shared resources. It presents the primary programming interface for the RS/6000 Distributed System in the form of system calls. Because the kernel is shared among all processes, any process running "in the kernel" (that is, running in privileged hardware state as the result of a context switch) is able to directly reference the data structures that implement shared resources.

The major components of the kernel are memory management, process management, the file system, the I/O system, and the network protocols (IP, TCP, UDP, and NFS).

5.2.4 Kernel Extensions

Kernel extensions are dynamically loaded code modules that add function to the kernel. They include device drivers, virtual file systems (e.g., NFS), inter process communication methods (e.g., named pipes), networking protocols, and other supporting services. Kernel extensions can be loaded only at system boot in the evaluated configuration.

Kernel extensions run with kernel privilege, similarly to kprocs. However, extensions differ from kprocs in that the kernel does not schedule them. Instead, kernel extensions are invoked from user processes by system calls, or internal calls within the kernel, or started to handle external events such as interrupts.

Kernel extensions run entirely within the kernel protection domain. An extension may export system calls in addition to those exported by the base AIX kernel. User-domain code can only

access these extensions through the exported system calls, or indirectly via the system calls exported by the base kernel.

Device drivers are kernel extensions that manage specific peripheral devices used by the operating system. Device drivers shield the operating system from device-specific details and provide a common I/O model for user programs to access the associated devices. For example, a user process calls ***read*** to read data, ***write*** to write data, and ***ioctl*** to perform I/O control functions.

5.2.5 Kernel Processes (kprocs)

The RS/6000 Distributed System has some processes that operate solely within the kernel on behalf of kernel entities. Kernel processes (kprocs) are processes that spend all of their time in supervisor state, running kernel code. There are no separate binaries associated with kernel processes, they simply exist as part of the kernel proper. They can be created only by the kernel and the only tasks they perform are on the behalf of kernel entities. They have a private u-area and kernel stack, but share text with the rest of the kernel. They are scheduled and selected for dispatch the same way as user processes and are designated within the process table as "kproc" and are single-threaded[1]. On creation the kernel sets a priority for kprocs. The kprocs of the RS/6000 Distributed System do not present a TCB interface because they are internal to the kernel, cannot be created by user processes, cannot use shared library object code from the user domain, and cannot be affected by user signals.

5.2.6 Trusted Processes

A trusted processes in the RS/6000 Distributed System is any process running with a distinguished user ID, distinguished group ID, or in an operating environment where it affects the correct operation of other such processes. Most high-level TCB functions are performed by trusted processes particularly those providing distributed services.

A trusted process is distinguished from other user processes by the ability to affect the security policy. Some trusted processes implement security policies directly (e.g., identification and authentication) but many are trusted simply because they operate in an environment that confers the ability to access TCB data (e.g., programs run by administrators or during system initialization).

Trusted processes have all the kernel interfaces available for their use, but are limited to kernel-provided mechanisms for communication and data sharing, such as files for data storage and pipes, sockets and signals for communication.

The major functions implemented with trusted processes include user login, identification and authentication, batch processing, audit data management and reduction, printer queue

[1] There is one exception to the single-threaded rule on the evaluated configuration. The "**GIL**" kproc is multi-threaded to handle the various timers that it must coordinate.

management, network file transfer, backup and restore, system initialization, and system administration.

5.2.7 User Processes

The RS/6000 TCB primarily exists to support the activities of user processes. A user, or non-TCB, process has no special privileges or security attributes with respect to the TCB. The user process is isolated from interference by other user processes primarily through the CPU execution state and address protection mechanisms, and also through DAC protections on TCB interfaces for process manipulation.

5.2.8 TCB Databases

Tables 5-1 and 5-2 identify the primary TCB databases used in the RS/6000 Distributed System and their purpose. These are listed both as individual files (by pathname) or collections of files. With the exception of databases listed with the User attribute (which indicates that a user can read, but not write, the file), all of these databases are only accessible to administrators.

When the system comprises more than one machine, the administrative databases are shared between the hosts. A single master copy of the databases is maintained on an administrative server and made available for NFS-mounting by all the hosts in the system. A particular user or group has the same identity on all hosts in the system. Some databases are maintained independently on each host in the system (Table 5-1), while others are synchronized through sharing (Table 5-2).

Table 5-1. Administrative Databases. *This table lists other administrative files used to configure the TCB.*

Database	Purpose
/etc/filesystems	Defines characteristics of mountable file systems.
*/etc/security/audit/**	Audit configuration files not specified in Table 5-1
/etc/security/failedlogin	Lists last failed logins for each user.
/etc/security/lastlog	Stores time/date of last successful login for each user.
/etc/security/login.cfg	Defines attributes enforced when logging in or changing passwords.
/etc/security/portlog	Records ports locked as a result of login failures.
/etc/security/roles	Defines administrative roles. Not used in the evaluated configuration.
/etc/security/smitacl.group	Defines which groups can use which Web-based System Management and SMITTY screens. Not used in the evaluated configuration.
/etc/security/smitacl.user	Defines which users can use which WSM and SMITTY screens. Not used in the evaluated configuration.
/etc/security/sysck.cfg	Defines file permissions, owner and group, and file checksum for verifying that TCB software has not been tampered with.
/etc/security/user.roles	Defines which users are permitted to assume administrative roles. Not used in the evaluated configuration.
/etc/vfs	controls another list of file system commands
/etc/inittab	contains commands to srcmaster daemon that starts other system daemons.

Table 5-2. Administrative Databases. *This table lists the shared administrative files used to configure the TCB.*

Database	Purpose
/etc/passwd	Stores user names, UIDs, primary GID, home directories for all system users.
/etc/group	Stores group names, supplemental GIDs, and group members for all system groups.
/etc/hosts	Contains hostnames and their address for hosts in the network. This file is used to resolve a hostname into an Internet address in the absence of a domain name server.
/etc/security/audit/bincmds	Specifies the pipeline of commands to be performed by the auditbin daemon.
/etc/security/acl	Specification of TCP port, host (or subnet), and user/group at that host or subnet allowed access to the port.
/etc/security/audit/config	Specifies who and what is going to be audited, where the bin audit data will reside, and how auditing will be performed.
/etc/security/audit/events	Defines all of the audit events that are recognized by the system and the form of their tail data.
/etc/security/audit/objects	Specifies file system objects whose access is to be audited along with for what access modes it will be done.
/etc/security/audit/streamcmds	Specifies the pipeline of commands to be connected to */dev/audit*.
/etc/security/environ	Stores default values for environment variables to be set at login time.
/etc/security/group	Provides additional information about AIX groups.
/etc/security/.ids	Defines the next available user and group id numbers to be used when creating new ids.
/etc/security/limits	Establishes resource consumption limits (memory, CPU, disk.).
/etc/security/passwd	Defines user passwords in one-way encrypted form, plus additional characteristics including previous passwords, password quality parameters.
/etc/security/.profile	Default profile to be used when a new user is created.
/etc/security/services	Specification of service names to be used by **DACINET** in the style of */etc/services*.
/etc/security/user	Defines supplementary data about users, including audit status, required password characteristics, access to su command.

5.2.9 Internal TCB Protection Mechanisms

All kernel software has access to all of memory, and the ability to execute all instructions. In general, however, only memory containing kernel data structures is manipulated by kernel software. Parameters are copied to and from process storage (i.e., that accessible outside the kernel) by explicit internal mechanisms, and those interfaces only refer to storage belonging to the process that invoked the kernel (e.g., by a system call).

Functions implemented in trusted processes are more strongly isolated than the kernel. Because there is no explicit sharing of data, as there is in the kernel address space, all communications and interactions between trusted processes take place explicitly through files and similar mechanisms. This encourages an architecture in which specific TCB functions are implemented by well-defined groups of processes.

5.3 TCB Software Components

This section describes the security-relevant aspects of each of the following software components that comprise the TCB software running on the system:

- Memory Management
- Process Management
- File System and I/O
- I/O Management
- Network File System
- Import and Export
- Backup and Restore
- Inter-Process Communication
- Low-Level Network Communication Protocols
- Network Application Protocols
- Identification and Authentication
- Interactive Login and Related Mechanisms
- Batch Processing
- Printer Services
- Mail
- Audit Management
- Initialization and Shutdown
- TCB Support

5.3.1 Memory Management

Memory management functions are performed by the Memory Management subsystem of the base kernel, referred to as the Virtual Memory Manager (VMM). The VMM is responsible for managing the system's physical and virtual memory resources. The virtual memory resources are implemented through paging.

A process makes references to memory using an effective address. The effective address is translated into a virtual address, which is used to resolve the physical address in memory. Address translation is described in section 4.6.2, Address Translation. The memory management hardware computes the virtual and physical addresses when the processor executes a load, store, branch or cache instruction, or when it fetches the next instruction.

5.3.1.1 Segmentation

AIX provides a segmented virtual address space for both the 32-bit and 64-bit processors. The virtual memory design gives the user programs the appearance of one large contiguous address space, even though the space used by the various programs is usually larger than the physical memory available on the system. This is accomplished by keeping the virtual memory that has

been recently referenced in physical memory, while virtual memory that has not been recently referenced is kept on a paging device.

The segmented architecture divides a process' address space into 256 MB pieces as an optimization for addressing. This approach provides a larger address space than permitted by straightforward mapping of 32-bit addresses, and allows the kernel to map different types of memory objects into each segment. Section 5.3.2.1, Typical Process Address Space contains a description of how segments are used by a process.

The segment protection keys, Supervisor and Problem State, are combined with the page protection bits to determine accessibility of a page. This access check protects user or kernel mode software from reading or writing particular memory addresses. A description of segment and page protection as it is implemented in hardware can be found in sections 4.6.4.1, Segment Protection and 4.6.4.2, Page Protection.

5.3.1.2 Paging

When a memory address not currently stored in physical memory is referenced, a page fault will occur. The process of retrieving the page from paging space and placing it in real memory results in the page being available for use. The user process only attempts to access an area of memory - the VMM makes that piece of memory available transparently. When the VMM loads a piece of virtual memory from disk, it will receive the referenced page, as outlined in figure 5.1 and the description below. When a new page is allocated by the VMM, the kernel fills the page with zeroes before making it available to the user process.

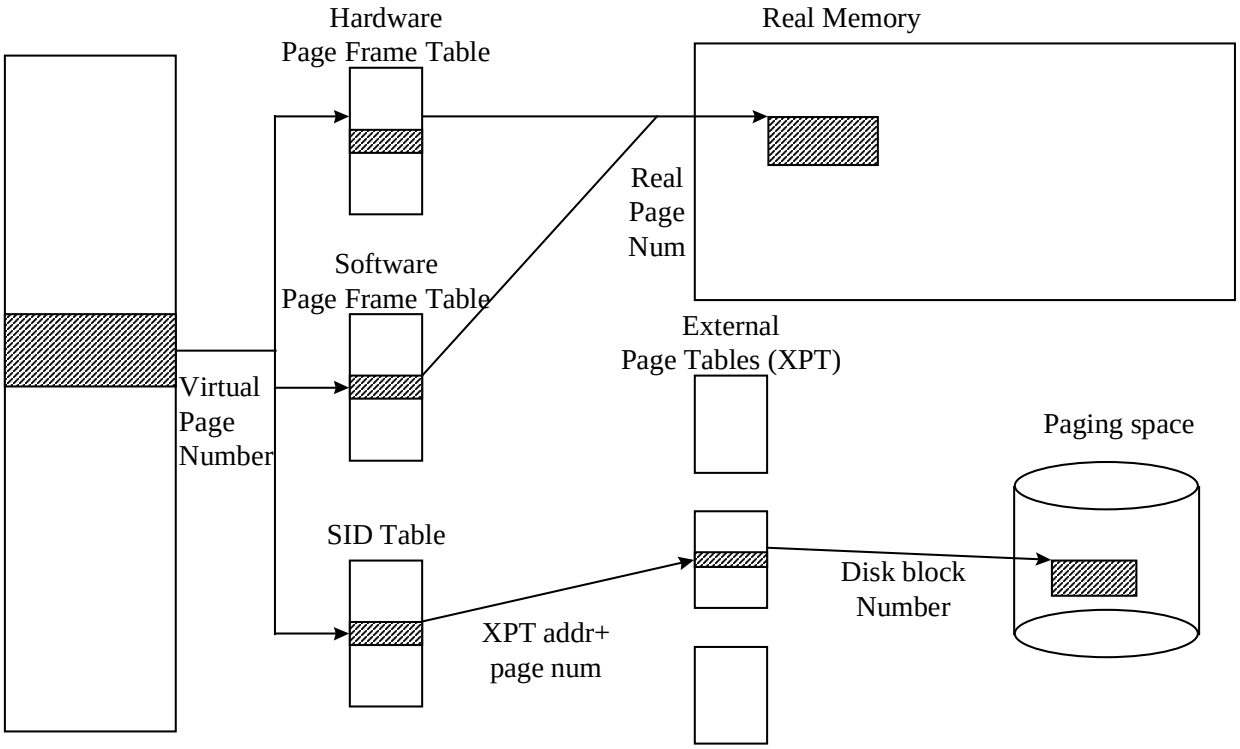

Figure 5.1: Page Frame Mapping

The software page frame table keeps track of all pages currently in use. The hardware page frame table keeps track of all the pages that are currently loaded in real memory. If the number of pages in use exceeds the amount of real memory, the software page frame table will be used in the page fault process to determine the location of a page. The hardware and software page frame tables are only available in the kernel's memory, while operating in kernel mode.

The process of mapping a page from real memory or paging space is as follows:

1. If the Translation Look-aside Buffer (TLB) has the mapping between virtual page number and physical page, then the hardware directly calculates the physical address and accesses the memory. Otherwise, continue at step 2.
2. The hardware consults the hardware page frame table (HPFT). If the page is found then the physical address is calculated and memory is accessed. Otherwise the hardware generates an exception which causes the VMM to be invoked in step 3.
3. The VMM looks in the software page frame table (SPFT) to determine if the page is in memory. If so, the VMM accesses the real page number, which reloads the page reference in the TLB and HPFT. User program execution is restarted and address translation recommences at step 1 above.
4. If the page is not in the SPFT, the VMM checks the segment ID table's segment control block for the referenced segment. The block contains a pointer to the external page table (XPT), which tells the VMM where to find the page on the disk. The XPT is a two-level table stored in a double-indirect format. Each of the two levels has 256 entries, for a total of 65536 page numbers. (65536 * 4 Kbytes = 256 Mbytes, the size of a segment). The VMM accesses the XPT for the segment, user program execution is restarted and address translation recommences at step 1 above.

5.3.1.3 Memory Protection and the Kernel

The page protection bits are generally set to the same values for each page in a segment, and are treated as a segment level attribute. The kernel has an internal interface to modify the page protection bits for a page range. This interface is used to modify the page protection bits for the segment.

Two exceptions exist where an individual page's protection bits would be changed to a value different from the remainder of the segment: the SVC table and I/O operations. The SVC table is described in section 5.3.2.1, Typical Process Address Space. Pages used in I/O operations may be hidden to prevent inadvertent access to them when the kernel is in the process of filling the page with data.

5.3.1.4 Pinned Memory

A portion of the address space available on the system may be pinned from within the kernel. When memory is pinned, the pager is unable to swap out the page that contains the pinned memory. A value is stored within the page frame table that is incremented when a memory address within that page is pinned. When the pager is searching for pages to swap out, it first

checks this field to see if a page is pinned. If this value is non-zero, the pager continues searching for a page to swap out.

A portion of the kernel's address space is pinned. Device drivers execute as kernel extensions, and are broken down into two pieces: the top half and the bottom half. The top-half of the device driver executes in the kernel's process environment and does not need to be pinned, while the bottom half runs in the kernel's interrupt environment and is not allowed to page fault. There are other pieces of memory that are pinned: the kernel's heap, the process table, the CSA (Current Save Area) and context save areas, all of the kernel's interrupt handling code, and the hardware and software page frame tables.

The interfaces for pinning and unpinning of memory addresses are implemented using kernel services, and are only available from within the kernel. The kernel services are internal to the kernel and provide no external interfaces.

The kernel may pin portions of the users address space, including the process's user area. The user area is discussed in section 5.3.2.3, Process Context. Device drivers reference the other portions of a user's address space, which are pinned and unpinned by the kernel, to pin/unpin user memory involved in DMA operations.

5.3.2 Process Management

A process is a program that is in some state of execution. A process consists of the information in the program along with information that the system requires for its execution.

Processes have a parent-child relationship. The first processes created on the system are the kernel processes (kprocs) and then INIT. Every other process invoked on the system is created using the ***fork*** system call, and is a child of the process that created it.

5.3.2.1 Typical Process Address Space

AIX Version 4.3.1 TCSEC Evaluated C2 Security supports 32-bit processes on the 604e, and 32 and 64-bit processes on the RS64. Each process contains a collection of segments that define the process address space. A process cannot access arbitrary locations in the virtual memory space because all references are made using effective addresses. An effective address will always be translated to a virtual address. The translation requires the use of the segment registers (32-bit, 32-bit on 64-bit processor) or the segment table (64-bit). User mode processes are constrained to the segments assigned by the kernel, because the instructions to move to and move from segment registers are supervisor mode instructions.

The kernel segment is mapped as Segment 0 of every process. The system call table (SVC) and system call initial code are the only portions of the kernel segment that are read-only to user mode processes. The kernel uses page protection to make the user viewable portions of the kernel segment read-only, and makes the remainder of the segment (the kernel text and data) invisible to user processes.

Table 5-3. Segments Contained in a Typical 32-bit Process. *Each 32-bit user mode process has sixteen 256MB segments available for use.*

Segment	Segment Name
0x0	Kernel segment (only parts visible)
0x1	User Text
0x2	Process Private
0x3-0xC	Available for user (shmat, mmap)
0xD	System Shared Library Text
0xE	Available for shmat or mmap
0xF	Per-Process Shared Library Data

Table 5-4. Segments Contained in the Typical 64-bit Process. *Each 64-bit user mode process has a theoretical 2**32 number of 256MB segments available for use. In implementation, the number of segments is limited to 2^{20}.*

Segment(s)	Segment Name
0x0	Kernel segment (only parts visible)
0x1	Kernel segment 2 (only parts visible)
0x2	Reserved for user mode loader
0x3-0xC	Available for user (shmat, mmap)
0xD	Reserved for user mode loader
0xE	Available for user (shmat, mmap)
0xF	Reserved for user mode loader
0x10 - end of text	User Text
end of text - end of data	User Data
end of data - end of BSS	User data that has not been initialized
end of BSS - end of heap	User Heap[2]
end of heap - 0x6FFFFFFF	Available for user (shmat, mmap)
0x70000000 - 0x7FFFFFFF	Default User shmat, mmap area
0x80000000 - 0x8FFFFFFF	User explicit load area
0x90000000 - 0x9FFFFFFF	Shared Library Text and Data
0xA0000000 - 0xEFFFFFFF	Reserved for system use
0xF0000000 - 0xFFFFFFFF	User Initial Thread Stack

The user text segment contains the program text, loaded from an object file by the ***exec*** system call. User mode processes have read-only access to this segment. This allows the kernel to maintain one segment with a particular program's code, and share it between multiple users.

The shared library text segment contains a copy of the program text for the shared libraries currently in use on the system. Program text for shared libraries is loaded at the low end of the segment. Shared libraries are automatically added to the users address space by the loader, when a shared-library is loaded. User mode programs have read-only access to this segment, to facilitate sharing between different user processes.

Shared library data segments are used to store the corresponding data associated with shared library text segments. Shared library data segments are not shared with other processes.

[2] This data is stored in segment two for a 32-bit process.

Shared data segments provide a method for a process to create and attach shared memory, which is accessible by other processes. Section 5.3.7, Inter-Process Communication describes shared memory and its restraints.

Memory mapped segments provide user and kernel mode programs the ability to map whole files into a segment. This is done as a performance enhancement, to allow loads and stores of a file to be performed in a user's virtual address space. DAC checks with the ***mmap*** system call are performed before the file is made available in a segment. The page protection bits for the pages that contain the memory mapped file are set depending on whether the user has read-only or read-write access.

The process private segment contains the user data, the user stack, the kernel per-process data, the primary kernel thread stack, and per-process loader data. This segment is not shared with other processes.

5.3.2.2 Kernel Process Management

The kernel process management function manages both processes and threads. It creates and destroys processes and schedules threads for execution. In an SMP computer, each CPU has its own scheduler and makes its own independent scheduling decision, based on shared data structures.

The process table is kernel controlled and lists all the currently running processes on the host. It is pinned in memory so that access to it is available at all times. Each process has an entry in the process table. The process table entry contains information necessary to keep track of the current state of the process. This table is located in the kernel extension segment, and is not accessible to user mode software.

The system may include multiple computers, but process management is a local function. There is no mechanism for a host to dispatch processes or threads across the LAN onto other hosts. Process IDs (PIDs) and Thread IDs (TIDs) refer to processes running on the local host's CPU(s), and are unique at any given time on a particular host. Different hosts may have the same PIDs and TIDs at the same time. PIDs and TIDs may eventually be reused on the same host.

Table 5-5. Process States. *The current state of a process determines whether the process is ready to execute or is waiting for something else to happen.*

State	Description
Idle	Temporary state while the ***fork*** call is allocating resources for the creation of the new process
Active	Normal process state
Stopped	Process has been stopped by the SIGSTOP signal
Swapped	Process cannot run until the scheduler makes it active again.
Zombie	Process was terminated and is awaiting final cleanup.

5.3.2.3 Process Context

Each process is assigned a process ID upon creation. The process ID provides a method for referencing the process, as well as an index into the process table.

The structures that define a process are the process table entry, the user area, and the process credentials. Additionally, each thread in the process has a structure that provides information about the thread and includes placeholders for register values when a context switch occurs.

The process table entry for a particular process stores a wide range of information about the process.

Table 5-6. Process Table Fields. *The following fields are a subset of the fields contained in a process table entry. These are derived from /usr/include/sys/proc.h which is shipped on all systems.*

Heading	Field	Description
Process	*p_stat*	Process state
	p_xstat	Exit status for wait
Process Link Pointers	*p_child*	Head of list of children
	p_siblings	NULL terminated sibling list
	p_uidl	Processes with the same p_uid
Thread	*p_threadlist*	Head of list of threads
	p_threadcount	Number of threads
	p_active	Number of active threads
	p_suspended	Number of suspended threads
	p_terminating	Number of terminating threads
Scheduling	*p_nice*	Nice value for CPU usage
Identifier	*p_uid*	Real user identifier
	p_suid	Set user identifier
	p_pid	Unique process identifier
	p_ppid	Parent process identifier
	p_sid	Session identifier
	p_pgrp	Process group leader process id
Miscellaneous	*p_auditmask*	Audit mask
	p_adspace	Segment ID for the Process private segment

The user area of a process contains information about the process that does not need to be in memory when the process is swapped out. Each process' user area is stored in its process private segment. The process table entry contains the value of p_adspace, which points to the segment ID for the process private-segment. Since the process user area may be swapped out, the process uses p_adspace to locate it.

The user area of the process cannot be modified by user-mode code, and is protected using the page memory protection mechanisms.

Table 5-7. Relevant User Area Fields. *The following fields are a subset of the user area fields. These are derived from /usr/include/sys/user.h which is shipped on all systems.*

Field	Description
U_ufd	User's file descriptor table
U_auditstatus	Auditing resume or suspend for this process
U_cred	User credentials
U_cmask	Mask for file creation
U_adspace	Array that stores the segment register values for the process
U_segst	Shared memory and mapped file segment information

The credentials structure contains all the relevant identification information for the process. This structure is pointed to from the user area, using the field U_cred.

The information stored in this structure is the basis for access control decisions and accountability on the system. Section 7.3, Discretionary Access Control, describes how the DAC mechanism performs access control decisions.

Table 5-8. Credentials. *The following fields are the complete credentials structure for each process. These are derived from /usr/include/sys/cred.h which is shipped on all systems.*

Field	Description
cr_ruid	Real user id
cr_uid	Effective user id
cr_suid	Saved user id
cr_luid	Login user id
cr_gid	Effective group id
cr_rgid	Real group id
cr_sgid	Saved group id
cr_mpriv	Maximum privileges
cr_ipriv	Inherited privileges
cr_epriv	Current privileges
cr_bpriv	Bequeathed privileges
cr_ngrps	Number of groups in group set
cr_groups	Group set list

5.3.2.4 Thread Context

Thread context consists of two structures: thread and uthread. The thread structure consists of per-thread information that can be used by other threads in the process. The thread structures make up the thread table, and are stored in the kernel extension segment. The uthread structure contains private per-thread data. The uthread structures are stored in the process private segment. Both the thread and uthread structures are inaccessible by untrusted software.

Table 5-9. Thread Table Fields. *The following fields are a subset of the fields contained in a thread table entry. These are derived from /usr/include/sys/thread.h which is shipped on all systems.*

Heading	Field	Description
Thread	*t_state*	Thread State
	t_flags	Thread flags
	t_tid	Unique Thread ID
	t_wtype	What the thread is waiting for
Signal	*t_sig*	Set of pending signals
	t_sigmask	Set of signals blocked by thread
Miscellaneous	*t_procp*	Pointer to owning process
	t_stackp	Stack Pointer

Table 5-10. Uthread Fields. *The following fields are a subset of the fields contained in a process table entry. These are derived from /usr/include/sys/uthread.h which is shipped on all systems.*

Heading	Field	Description
Uthread	*ut_save*	Machine state save area
	ut_msr or ut_msr64	Machine status register value

A process consists of at least one thread. Process context provides the environment necessary for a thread to execute. The thread contains the executable content and scheduling information. A process provides the following context to the thread: process attributes, address space, environment, working directory, file descriptor table and signal actions.

A thread references the process context to determine the values for the segment registers that make up the process address space. Each thread keeps its own copy of the registers, stack, scheduling priorities and pending/blocked signals.

Threads are visible to other processes using the **PS** command. Threads have no external interfaces that can be invoked by other untrusted processes.

5.3.2.5 Process Creation

A new user process is created when its parent makes a ***fork*** system call. The ***fork*** routine makes an almost exact copy of the parent's process context and passes it to the child. The differences between the parent and child process are the values stored for the process id and the parent process id. The resource statistics of the child process are set to zero when the ***fork*** occurs

The child inherits the following items from the parent:

- priority and nice values
- credentials
- process group membership
- the parent's opened files just before the ***fork***
- environment, signal handling settings
- SUID and SGID mode bits
- attached shared libraries
- attached shared memory segments
- attached mapped file segments

5.3.2.6 Process Destruction

The ***exit*** system call is used to terminate an existing process. The first task in the termination process is for the kernel to adjust the process table, modifying the process being terminated to zombie status. All files held open by the process are closed, and the VMM deallocates the memory segments held by the process. The parent receives a signal to make it aware that the child process has terminated.

A signal can also be the cause of process destruction. If a process attempts to access a memory location that it does not have access to, an exception will be generated, and a signal will be sent to the process to terminate it. Section 5.3.7, Inter-Process Communication discusses signals as an IPC mechanism.

5.3.2.7 Program Invocation

The ***exec*** system call is used to execute a new program in the calling process. The ***exec*** system call does not create a new process, but overlays the current process with a new program, which is called the new-process image. The new-process image maintains its file descriptor table and user area during the ***exec*** system call. The new process image may result in the change of identity, depending if the process is set-user-ID or set-group-ID.

A set-user-ID program sets the effective user-id of the user invoking the program to the UID of the owner of the program. A set-group-ID program sets the effective group-id of the user invoking the program to the GID of the owning group of the program. This allows a process to execute using the permissions of the owning user or group. A program is marked as set-user-ID or set-group-ID using bits stored in the program's inode on disk.

5.3.2.8 Multiprocessing

The kernel provides mechanisms to ensure that in the multiprocessing environment two threads do not inadvertently overwrite data simultaneously. The kernel supports this through serialization and locks. An additional mechanism called funneling is used with regards to device drivers in the multiprocessing environment. The multiprocessing hardware maintains the consistency of the system caches, as described in section 4.5, Multiprocessing.

Funneling is the process of limiting a device driver to execution on the master processor. A single processor is initially in control during the boot process. This first processor is designated as the master processor. Device drivers that are not labeled as MP safe or MP efficient will only run on the master processor. Device drivers that are MP safe provide correct data serialization in the MP environment. Device drivers that are MP efficient are MP safe, and the design is optimized for efficiency in the MP environment.

Serialization is the mechanism of arranging for processes to access data serially, instead of in parallel. Before a program modifies a shared data item, it must ensure that no other thread will change the item. Serialization is implemented using locks. A lock represents permission to access one or more data items. Lock and unlock operations are atomic, so neither interrupts or multiprocessor access affect their outcome. A thread that wishes to access shared data must acquire the lock to that data before accessing it.

There are two different types of locks: simple and complex. Simple locks provide exclusive ownership to a shared data item. Complex locks provide read/write access to a shared data item.

5.3.3 File System and I/O

This section describes the RS/6000 Distributed System file system, which provides access to information stored in file system objects (FSOs). This section begins with a discussion of the file system objects, followed by a description of the file system implementation and concludes with a discussion of each file system type allowed in the evaluated configuration.

5.3.3.1 File System Objects

The RS/6000 Distributed System file system is organized as a hierarchy of directories that contain file system objects (FSOs). File system objects include directories, ordinary files (which may contain programs or data), symbolic links, and various special types of objects. At system startup, the root file system is mounted. The root file system contains a number of directories that serve as mount points where other file systems are attached to create the full file system hierarchy. During system initialization a standard set of file systems is mounted, see section 5.3.16, Initialization and Shutdown for further details.

All FSOs have a common set of security attributes that can be modified and examined. See section 6.4, File System Resources Security Attributes for further details. Some FSO types have additional security attributes, described specifically for each object type. There are several types of FSOs. These are distinguished by the operations that may be performed on them as well as by the semantics of some operations.

5.3.3.1.1 Directories

Directories are FSOs that organize the file system into a hierarchy. Each directory can contain other FSOs, including other directories (a directory is actually a file that contains pointers to the FSOs "contained" in the directory). Each FSO is named by a pathname, which is a sequence of directory names followed by the FSO's name. The names in a pathname are separated by slash ("/") characters. A pathname may start with a slash, indicating that it is an absolute pathname and is interpreted starting from the root directory, which is at the top of the hierarchy. If a pathname does not start with a slash, it is interpreted relative to the current directory of the process. Each directory contains two special names, one of which (".") refers to the directory itself, and the other ("..") refers to the containing directory.

5.3.3.1.2 Ordinary Files

The most common FSOs are ordinary files: they are opened and closed, and while open, data can be read and written. These operations, as well as device-specific ***ioctl*** operations, are also supported for device special files and, with the exception of writing, by directories. Regular files can also be used to store executable programs.

5.3.3.1.3 Links

There are two types of links: symbolic and hard.

Symbolic links are used to organize the file system. A symbolic link contains a pathname, called the target. When a symbolic link's name is encountered by the kernel while parsing a pathname, the contents of the symbolic link are logically prefixed to the remainder of the pathname, and the parsing is restarted. A symbolic link in one directory can reference an object in an entirely different location in the file system, and permit different pathnames to refer to the same object.

A hard link is not an FSO as such, but is a way of referring to the same object with multiple names. Unlike a symbolic link, which has no restrictions on the new pathname it supplies (the target need not even exist), a hard link is a name in a directory that is created referring to a specific FSO and has restrictions on scope (e.g., it cannot name an object located in another file system). The number of hard links pointing to an FSO is recorded within the FSO. The object will not be deleted as long as a hard link to it exists.

5.3.3.1.4 Pipes

Pipes are an inter-process communication mechanism described in detail in section 5.3.7, Inter-Process Communication. There are two types of pipes: unnamed and named. Unnamed pipes do not have file names but are considered part of the file system because they are manipulated by users through the file system interfaces, and are controlled within the kernel by file system structures. Unnamed pipes do not have permanent storage. Transient storage, in the form of buffers, is allocated from the kernel heap. An unnamed pipe is deleted when all processes that have a file descriptor release the file descriptor or terminate.

Named pipes implement data transfer using the same mechanisms as unnamed pipes, but they exist as actual FSOs that must be opened to be used. Like unnamed pipes, FIFOs do not have permanent storage, but they do have pathnames, and must be created and deleted explicitly. FIFOs are visible across NFS but cannot be read or written between different hosts. The two communicating processes must reside on the same host.

5.3.3.1.5 Device Special Files

Device special files provide a common interface to physical and virtual devices allowing them to be manipulated in the same manner as regular data files. Device special files are named by pathnames and accessed like files, supported by the SPECFS file system.

Devices in AIX are implemented as a hierarchy of logical devices each providing a virtual device with certain operations. Each of these devices, in addition to the physical device, has a device special file, which contains pathname data that the device driver uses to find other logical devices. Ultimately a device special file for a logical device points to the device special file for the appropriate physical device.

Device special files are created by the administrator and can only exist in the */dev* directory as mandated by the TFM.

5.3.3.2 File System Implementation

The AIX file system is implemented in three layers: Logical File System (LFS), Virtual File System (VFS) and Physical File System.

5.3.3.2.1 Logical File System

The logical file system (LFS) is the layer of the file system where users can request file operations by system calls such as ***open, close***, and ***read***. This layer provides a common interface to the user irrespective of the underlying file system implementations (e.g. it is irrelevant if the file is a local JFS file, or a remote NFS file). This level of the file system manages the user open file descriptor tables and the system open file table.

The user open file descriptor table maintains a record of the files that a process currently has open. This table is located in the process user area. When a program is started the first three entries in the table are the default open files, standard in (*stdin*), standard out (*stdout*), and standard error (*stderr*). These three file descriptors are associated with the input from, and output to, the session's controlling terminal. A process may decide to close those file descriptors (as is commonly done by daemons). At that time they may be reused unless they are privileged programs (SUID/SGID) at which point they are associated with */dev/null*.[3] When a process opens files an entry is created in the user open file descriptor table. The process then refers to the file by

[3] This unique, non-POSIX, behavior is as a result of a known exploit.

its relative position in this table. The file entry in the user open file descriptor table points to an entry in the system open file table.

The system open file table holds information for all files open on the system and is used to manage I/O activity to these files. There is a separate entry in the system open file table corresponding to each open operation. Initially, this corresponds to exactly one file descriptor in one process, but if the file descriptor is duplicated with ***dup***, or shared with a child over a ***fork***, several file descriptors may correspond to the same system open file table entry. Entries in this table contain several fields holding data describing the current I/O condition of each file:

- status flags - indicating the mode in which the file was opened (e.g. read, read/write)
- file vnode location - discussed later in this chapter,
- file reference count - indicating how many processes are sharing the open file, and
- current byte offset into the file where the next I/O operation will occur.

5.3.3.2.2 Virtual File System

Below the LFS layer, the VFS layer provides applications with access to different types of physical file systems without requiring users or applications to know the specifics of how the file systems are implemented.

The evaluated configuration supports four types of virtual file systems:

- journaled file system (JFS),
- network file system (NFS),
- CD-ROM file system (CDRFS), and
- special file file system (SPECFS) - this is used internally by the kernel to support the device I/O subsystem.

In the standard AIX product other types of file systems can be added through kernel extensions. The evaluated configuration is restricted to the four mentioned above. The TFM forbids the administrator from introducing other kernel extensions not present on the evaluated configuration installation media.

There are four structures that support the VFS: the vnode and gnode, which represent files; and the VFS structure, and GFS structures which represent file systems.

5.3.3.2.2.1 File Structures

A virtual node or vnode represents access to an object within a virtual file system. Vnodes are used to translate a path name into a generic node or gnode. A vnode is either created, or used again, for every reference made to a file through a path name. When a user attempts to open or create a file, if the VFS structure containing the file already has a vnode representing that file, a use count in the vnode is incremented and the existing vnode is used. Otherwise a new vnode is created. Vnodes for each directory searched to resolve a pathname are also created, or referenced. Vnodes are also created for files as the files are created.

Table 5-11. The vnode structure as defined in /usr/include/sys/vnode.h.

Field	Description
flag	indicates the status of the vnode (whether the associated file system is mounted for example)
count	the reference count
vfsp	pointer to the VFS structure
mvfsp	pointer to the VFS structure if this vnode is the mount point for the file system otherwise this is null
gnode	pointer to gnode structure

A gnode is the representation of an object in a file system implementation. There is a one to one correspondence between a gnode and an object in a file system implementation.

The gnode serves as the interface between the logical file system and the physical file system implementation. Calls to the file system implementation serve as requests to perform an operation on a specific gnode.

A gnode is embedded within the file system implementation specific structure (inode for JFS, rnode for NFS, cdrnode for CDRFS, and specnode for SPECFS). A gnode is needed in addition to the file system inode or its equivalent, because some file system implementations may not include the concept of an inode. The gnode structure substitutes for whatever structure the file system implementation may have used to uniquely identify a file system object. The logical file system relies on the file system implementation to provide valid data for the following fields in the gnode: the type field which identifies the type of object represented by the gnode and the operations field which identifies the set of operations that can be performed on the object. Gnodes are created as needed by file system specific code at the same time the inode (or equivalent) is created. This is normally immediately followed by a call to a kernel service to create a matching vnode.

Table 5-12. The gnode structure as defined in /usr/include/sys/gnode.h.

Field	Description
segment	segment into which the file is mapped
operations	pointer to vnodeops structure
vnode	pointer to vnode structure
type	the type of object represented

5.3.3.2.2.2 File System Structures

There is one VFS structure for each file system currently mounted. The VFS structure is central to each mounted file system. It provides access to the vnodes currently loaded for the file system, mount information, and a path back to the gfs structure and its file system specific subroutines through the vfs_gfs pointer. The VFS structures are a linked list with the first VFS entry pointing to the root file system.

Table 5-13. The VFS Structure as Defined in /usr/include/sys/vfs.h.

Field	Description
next	A pointer to the next VFS structure
gfs	A pointer to a gfs structure
mntd	A pointer to the vnode that represents the root directory of the file system of this vfs structure
mntdover	A pointer back to the vnode that represents the mount point
vnodes	All vnodes in this VFS
count	number of users of this VFS
mdata	mount information structure

The GFS structure contains information specific to each file system type. There is one GFS structure in the kernel for each supported virtual file system type. By default, the evaluated configuration has four GFS structures, one each for JFS, NFS, CDRFS and SPECFS. For each GFS entry, there may be any number of VFS entries. The GFS structures are stored within a global array accessible only by the kernel. The GFS structure includes the following fields:

Table 5-14. The GFS Structure as Defined in /usr/include/sys/gfs/h.

Field	Description
ops	A pointer to the vfsops structure
ops2	A pointer to a vnodeops structure
type	An integer that contains a description of the VFS type: NFS=2, JFS=3, and CDRFS = 5.

A GFS, vnodeops and vfsops structure are created each time a new file system kernel extension is added to the kernel. When new file systems are mounted, a VFS structure and mount structure are created. The mount structure contains specifics of the mount request such as the object being mounted, and the stub over which it is being mounted. The VFS structure is the connecting structure, which links the vnodes (representing files) with the mount, structure information, and the GFS structure.

5.3.3.2.3 Physical File System

Generally, a physical file system is a set of code that implements the vnode/ VFS interface and which stores the file data on a local disk. Each of the file virtual file system descriptions given below discusses the physical file system implementation for that particular file system type.

5.3.3.3 Virtual File System Types

There are four types of file systems supported by the RS/6000 Distributed System providing access to different types of information in a uniform manner. Each of these are discussed in the following sections.

5.3.3.3.1 JFS

The native file system type for AIX is the Journaled File System (JFS). JFS provides storage of files on locally connected disks. The JFS is similar to many other UNIX file systems supporting the entire set of file system semantics, providing an interface that is compatible with those file systems. The main difference is that JFS uses database journaling techniques to maintain its

structural consistency. This prevents damage to the file system when the system is halted abnormally. JFS uses a log replay mechanism to provide a method of recovery after a file system crash.

5.3.3.3.1.1 JFS Physical File System

The JFS physical file system organization is based on a segment. A segment is a logically contiguous, but physically noncontiguous, set of data blocks. Segments are described by a disk inode, which is a permanent record of the mapping of the logical blocks of a segment to physical blocks. The JFS file systems have the following components:

Table 5-15. Physical File System Components.

Field	Description
superblock	file system description including size allocation and consistency of on-disk data structures.
diskmap	bit map indicating whether each block on the logical volume is in use
disk inodes	all disk inodes for the file system
disk inode extensions	additional information (e.g., access control lists) which is too large for fixed size disk inode.
indirect blocks	list of indirect data blocks which compose a file (if the file is sufficiently large)
directories	the file system namespace

5.3.3.3.1.2 JFS Support Structures

The JFS structures are the same as those described under the physical file system implementation. The superblock exists at a well known disk address and records information global to the file system. The remainder of the file system space is devoted to disk inodes and data blocks.

The disk inode is the primary structure supporting all FSOs in a JFS file system. There is a one-to-one correspondence between a disk inode and a file. The disk inode contains the following information:

- File type
- File access times
- File permission
- File owner/group
- Number of links to the file
- Size of the file
- Number of disk blocks in file/directory

Since it would be inefficient to constantly perform a disk read to get the on-disk inode information, when a file is opened an in-core inode is created, or located if already resident.

The incore inode contains:

- forward and backward pointers to indicate its place in the hash queue
- the gnode structure
- disk inode number of the file
- device major and minor numbers for the file's file system
- a lock used by the kernel when it is updating the inode
- a reference count for the file (how many vnodes point to the gnode within this inode)
- the disk inode

All existing in-core inodes are kept in a hash table called the in-core inode table (ICIT), accessed by device and index. This ensures that multiple inodes are not created for the same FSO. If an FSO is currently in use its underlying device must be mounted.

5.3.3.3.1.3 Open, Close and Delete Operations

When an application attempts to open a JFS file it supplies the path name of the file and the requested access, such as read/write. The access check is performed at this time. For each component of the path, a vnode is found in memory or created as needed, the associated in-core inode is also either found in memory or created as needed by fetching the disk inode. Based on the access information contained within the in-core inode (both regular permissions and extended permissions, if applicable) and the process credential structure, the DAC algorithm (found in Chapter 7) is performed and if the access requested is permitted the open continues, otherwise the open fails.

If the access check is passed, an entry is made in the system open file table, the vnode associated with the gnode is either reactivated, or if already active, the vnode count is incremented, and the user file descriptor table is updated to reference this new entity.

If the gnode is not resident because its associated in-core inode has been aged from the ICIT, the disk inode is fetched from disk and the in-core inode is created and placed in the ICIT either occupying a vacant entry or by aging the least recently used entry. The vnode pointed to by the gnode is reactivated and the counter for the gnode contained within the in-core inode is incremented. A new entry is added to the system open file table and the user file descriptor table is updated to reference this new entry.

Finally, if the file was not already open, the VMM assigns a segment and maps the data blocks of the file to the pages of that segment.

5.3.3.3.1.4 Read, and Write Operations

Other file system operations are those that are performed on the data contained in a file and usually involve reading or writing the data in a file. However, before any data in a file can be accessed the file must be opened, a file descriptor obtained, and entries created in file management tables. A file descriptor is an integer index into one of these tables and serves as an identifier used by a process to read and write an open file's contents.

Files are read or written using the file descriptor that ultimately leads to the file inode containing the description of where the requested data is located on disk. This information is used, along with kernel memory buffers, to either read data in from disk and then deliver it to the process, or write data out to disk.

5.3.3.3.1.5 Directory and Link Operations

Certain operations may be performed on directories, such as the creation of new directories, removal of existing directories, and creation of links to FSOs. Directories are a special type of file containing directory entries. A directory entry is a pointer to an FSO that is contained in a directory. These FSOs can be files, other directories, or links to other FSOs. A directory entry consists of a symbolic name for the FSO (i.e., the file name or directory name), a link (i.e., the i-node number) to the object, and miscellaneous other information (including the directory entry length and next directory entry offset).

A new directory can be created within an existing directory. This causes a new directory file to be created and a new directory entry to be written into the parent directory file. Access checks are made to verify that the requesting process has the necessary access rights to the parent directory (and also that the new directory does not already exist).

The symbolic name contained in a directory entry can be replaced with another one to give a file a new name. Links in directories can be created as either hard links or symbolic links. A hard link consists of a new directory entry pointing to an existing FSO, and is not an independent object. A hard link is created when a directory is opened and a new directory entry pointing to an existing FSO is written. The inode of the FSO contains a link count that records the number of existing hard links to the object. A hard link may be created to any type of FSO except a directory or a symbolic link, and may only point to an FSO in the same file system as the hard link.

A symbolic link consists of a special data file that contains pathname information about a target file. The kernel uses the pathname information in the symbolic link file when processing a pathname lookup. The link file and its directory entry are created when a symbolic link is created. A symbolic link is an object by itself; it exists independently of its target, and there are no restrictions on what a symbolic link may point to. The contents of a symbolic link can be read; this returns the pathname information contained in the link file. The ability of a process to obtain this information is dependent only on the attributes of the containing directory.

5.3.3.3.1.6 Locking Operations

AIX provides data structures and a programming interface for implementing file and record locking. Locks are enforced in one of two ways: advisory locks and mandatory locks. Advisory locks are implemented between two or more cooperating processes. Before a cooperating process attempts to read from or write to a region of the file, it checks for the existence of a lock. If a lock already exists, the process can either sleep until the lock becomes available or return an error condition to the application. If the region is not already locked, the process can then set a lock on the region. The kernel guarantees that the lock test and set operation is performed atomically to prevent a race condition.

By default, AIX locks are advisory locks. The kernel will implement mandatory locks for any data file that has its SGID bit set and the group execute bit clear. Mandatory locks are enabled with the **CHMOD** command, by setting the SGID bit and clearing the group execute bit. A process uses the ***fcntl*** system call to request a file lock. This system call may either block or return immediately, possibly with an error indicating some requested portion of the file has been locked by another process. Mandatory locks do not require the co-operation of other processes.

The kernel maintains a table, called the lock list, for implementing file and record locking. When a process requests a lock on a portion of a file, the kernel inspects the requested file's inode for a pointer into this table. If one is found, the kernel follows the pointer(s) in the lock list that describes a lock for the same file to determine if the region of the file is already locked. If not, an unused record in the table will be updated with the new lock request information and is inserted into the doubly linked list.

Since only one process can lock a region of a file at a time, the child process does not inherit file and record locks from the parent process. All file and record locks owned by a process are released when the process closes the file or when the process terminates.

5.3.3.3.2 CDRFS

The CDRFS permits access to various formats of read-only file systems on CD-ROM media. No writable CD-ROM devices or device drivers are included in the evaluated configuration. Access to the contents of a CD-ROM is performed through the normal file system interfaces.

The TFM mandates that the permissions on the CDROM devices remain root-only. However, it is possible for permissions on individual files of a CDROM to be set when the CD is created. These permissions cannot be modified because a CD is a read-only device.

5.3.3.3.2.1 CDRFS Physical File System

The physical attributes of the CDRFS file system are defined using the ISO-9660: 1988 Volume and File Structure of CD-ROM for Information Interchange, the System Use Sharing Protocol (SUSP), and the Rock Ridge Interchange Protocol (RRIP). The ISO-9660 standard defines the format to which information on a CD-ROM disc must adhere. SUSP and RRIP extend the ISO-9660 standard and provide a mechanism for storing POSIX file information as a portion of each file. The AIX operating system supports the ISO-9660 CD-ROM format, as well as the SUSP and RRIP protocols.

The volume space of a CD-ROM is recorded as a sequence of logical sectors and is organized into logical blocks. The logical sector size is 2048 bytes. The logical block size is no smaller than 512 bytes.

The primary volume descriptor contains a collection of values about the volume. Security relevant values contained about the volume are the logical block size for the volume, the location of the root directory record, volume creation date and time, and the size and location of the path table. The path table is defined by the ISO-9660 specification, but is not used by the CDRFS to resolve pathnames.

Directories are files that contain pointers to data files or other directory files. From the root directory entry, there can be up to seven directory levels below. A directory record describes each directory or file. A directory record contains the location of the file/directory on the disc, a file identifier and a series of flags about the file. These flags determine whether the file is a directory or normal file.

When the kernel is searching for a file, it begins by finding the root directory entry. If the file being searched for is located in the root directory, the files location on disc is located using the root directory entry. If the pathname that is being resolved points to a file that is in a directory below the root, the kernel follows the root directory entry to the next level directory record, pointed to by the root directory entry. The directory entry pointers are processed one at a time, through the subsequent levels until the file is located or until the directory furthest from the root is reached. If the directory furthest from the root is reached and the file has not been located, the file requested does not exist.

ISO-9660 defines an Extended Attribute Record (XAR). The XAR contains fields for ownership by user and group, permissions on the file/directory, and statistics of when the file was created and modified. The XAR provides a limited scope of security values that the operating system enforces.

If an XAR exists for a file or directory, it can occupy multiple sequential logical blocks, beginning with the first logical block for the file/directory. The length of the XAR is specified in the directory record. If an XAR does not exist, the default values for ownership and access permissions are ownership by the root identity and group system, with access permissions set to octal 555. The kernel places these values in the CDRNODE after discovering that no XAR exists. If an XAR for the directory record exists, the values for owner, group and permissions are set by the XAR.

SUSP defines a set of generic fields to be used to extend the system use area (SUA) of the ISO-9660 directory records. The SUA is space allocated within the ISO-9660 directory record format that is not utilized by ISO-9660. This space was set aside to allow custom implementations on top of ISO-9660. System use fields (SUF) are variable length fields for defining extensions to the ISO-9660 directory records.

AIX recognizes and processes SUSP records, but these records are not security relevant. They are used to provide padding and continuation, and to split up the SUA into smaller pieces

RRIP provides a mechanism to specify POSIX file system information on read-only compact discs. Nine system use fields are defined by RRIP. These fields are contained within the SUA for a directory record.

The PX field is used to represent the POSIX file attributes. The RRIP PX definition allows for setuid and setgid programs to be contained on a CD-ROM. This capability is restricted through a TFM warning to the system administrator that mandates no CDROMS be mounted for use that contain setuid/setgid programs. Only administrators (i.e. users capable of using the root identity) are able to mount CDROM devices.

Table 5.16: RRIP System Use Fields.

System Use Field	Description	Mandatory/Optional
PX	POSIX File Attributes (using the method outlined in section 7.3.1 to represent Unix permission bits, in their octal form)	Mandatory
PN	POSIX Device Modes	Mandatory (for block/character devices)
SL	Stores the content of a symbolic link	Mandatory if PX is set for symbolic link
NM	Alternate name of file/directory, different from the ISO-9660 defined name for the directory record	Optional
CL	Records the new location of a directory that has been relocated	Optional (used in conjunction with PL and RE)
PL	Specifies the original parent directory of a directory that has been relocated	Optional (used in conjunction with CL and RE)
RE	Specifies that this directory has been relocated	Optional (used in conjunction with CL and PL)
TF	Records time stamp information	Optional
RR	Flags indicating which fields were recorded for this directory record	Optional

When a CD-ROM is created, the permissions on the files used as the source image are transferred to the CD-ROM media using the standard defined for ISO-9660 XARs or RRIP. Once the CD-ROM has been created, the permissions cannot be changed, as the compact disc media cannot be overwritten.

5.3.3.3.2.2 CDRFS Support Structures

As discussed earlier in the file system implementation, each file system must have a gnode structure. For CDRFS, the gnode structure is contained within the CDRNODE. The CDRNODE is equivalent to the in-core inode of JFS and contains the gnode for CDRFS files. The CDRNODEs are maintained within a cache managed by the kernel. As with the JFS, if the CDRFS FSOs are currently in use the underlying CDROM must be mounted.

Table 5.17. The CDRNODE structure.

Field	Description
gnode	pointer to the generic node
number	cdrnode number
dirent	directory entry address
gen	cdrnode generation number
mode	file permissions
uid	file owner identification
gid	file group identification
nblocks	number of logical blocks used

5.3.3.3.2.3 CDRFS Operations

When an application attempts to open a CDRFS file it supplies the path name of the file and the requested access, in this case read or read/execute. The access check is performed at this time

against the permissions contained within the CDRNODE. The CDRNODE is populated from the XAR record or the RRIP PX system-use fields as appropriate for the format type of the CDROM.

An order of precedence for CD-ROM security attributes is enforced by AIX. ISO-9660 directory records and SUSP fields do not contain security attributes, so they do not affect the order of precedence. If a CDROM did not include Rock Ridge or an XAR, the default values specified earlier in the section would be the permissions used. If a CDROM contained the Rock Ridge format, it would be used in place of the default values. If a CDROM contains an XAR, it would be used in place of the default values. If a CD-ROM properly follows the Rock Ridge format, it would not contain an XAR. However, if a Rock Ridge CD-ROM were not properly formatted and contained both an XAR and Rock Ridge information, the XAR will take precedence and would be using to populate the security attributes in the CDRNODE rather than the Rock Ridge information or the default values.

For each component of the path, a vnode is found in memory or created as needed, the associated CDRNODE is searched for in the cdrnode cache. If the access check is passed, a new entry is added to the system open file table and the user file descriptor table is updated to reference this new entry.

5.3.3.3.3 NFS

The Network File System (NFS) is a distributed file system that provides for remote file access to support the appearance that files stored on a remote machine are present in the local directory hierarchy. Remote access to files through NFS requires no special user action (unlike remote access to files through FTP, which requires users to transfer remote files explicitly before use). The same system calls that work on local files work on files manipulated through NFS.

Similarities between NFS and JFS file access are:

- File names are all in the hierarchical file system tree
- Standard system calls are used: ***open, close, read, write, stat, lseek, link, unlink, chmod***, and ***chown***.
- File access checking is identical in cases other than the file owner. (See Chapter 7 for the description of this exception).

Differences arise from distribution of access control:

- Access to a remote file is lost when the file is removed on the remote machine. The remote access cannot keep a link open to that file.
- Access to a remote file may be lost when the file's mode, owner, or group is changed. Access checks are performed each time an operation is done on the file.
- Modifications to the file may have a delayed effect for readers of the file because of network delays and distributed buffering.

The NFS server and client are both part of the TCB; the client is part of the kernel file system, and sends NFS requests in response to file and I/O related system calls issued by a local process, referred to as the client process. The client kernel completely identifies the client process through passing the process credential structure of the local user process in the RPC call to the NFS server

kernel. The NFS server kernel performs the operation (such as creating a directory, or reading a block from a file) based on the identity of the client process (identified through the credential structure passed from the client kernel) and sends the result to the client kernel. The client kernel then, in turn, sends a result back to the client process that issued the system call resulting in the NFS action. This process is completely analogous to what occurs in a system call resulting in a JFS specific operation except that two kernels are involved in answering the system call, both the client kernel where the client process exists and the NFS server kernel where the object exists.

5.3.3.3.3.1 Low-level NFS Implementation

NFS functions depend on the Remote Procedure Call (RPC) protocol to call the low-level file access functions on the remote host. RPC eliminates the need for fixed assignment of privileged ports to protocols. A client kernel requiring NFS services must first communicate with the server's portmap process or portmapper, giving it the name of the RPC service it wants. The portmapper returns the port number that implements the service, if the service is available. Refer to section 5.3.9.4, NFS for a complete description of the portmapper. The NFS server kernel satisfies client kernel requests through the NFS server VFS and JFS and returns the results of the access to the client kernel. The client kernel finishes the transaction by working up through the VFS to the LFS and eventually returns results, for example a file descriptor, to the client process.

5.3.3.3.3.2 NFS Server Implementation

The NFS server process is started in user space with one NFS daemon or nfsd. This nfsd may start one, or more, additional nfsds that open a TCP socket and pass the socket descriptor to the kernel via an ***ioctl***. If there are multiple nfsds, they are in a queue that waits for a request from an NFS client to arrive in the receive queue of the socket. The requests are serviced, in turn by the nfsd at the top of the queue. Multiple nfsds are started to enhance throughput performance on servicing requests. A complete description of nfsd and the RPC daemons supporting NFS (rpc.lockd, rpc.statd and rpc.mountd) is contained in section 5.3.9.4, NFS.

Each request serviced by an nfsd contains a file handle for the file being accessed. The file handle contains all the information required by the server JFS to locate a vnode and associated incore-inode for the requested object. If neither of these structures exist, the file handle is used to locate the file directly on disk and create and populate these structures from the information contained in the disk inode through the process described in the JFS description. Since NFS servers are stateless, there is no guarantee that these structures exist on subsequent service requests and each attempt to access the same file could repeat the procedure of fetching the disk inode from disk and creating and populating the vnode and in-core inode structures. Further information on the file handles is provided below. The NFS server completes the requested operation by accessing, or creating, the vnode in the server's local JFS and returning the results to the client kernel.

5.3.3.3.3.3 NFS Client Implementation

NFS client kernel operations are initiated via NFS specific vnode operations (i.e. ***vn_ops***) as a result of a client process issuing file and I/O system calls for objects that are NFS mounted. In some cases, the client process request can be satisfied without making a call to the server. An

example is a ***stat*** when the current file attributes are cached on the client and the cache has not expired, allowing the client kernel to return the results to the client process.

Client NFS operations are very similar to JFS operations. Determining access permissions to a file is done by accessing the NFS specific vnode operation that results in a system call to the client kernel. The client kernel access routine checks the cached file attributes to determine if access has been granted for this client process to this object previously. If it hasn't, access is denied until the client attribute cache is refreshed (at most 3 seconds) and no RPC request to the NFS server is made. If access has been granted, the client kernel issues an RPC request to the server kernel to check access. The information passed from the client kernel contains all the information required by the server kernel to determine whether access should be granted. This information includes, the credential structure[4] that identifies the user, the file handle that identifies the object, and a bit mask that identifies the mode requested (for example, read or read/write). Other operations work in a similar manner and are further described in section 5.3.9.4, NFS.

5.3.3.3.3.4 NFS Support Structures

Since NFS depends on both a client and a server there are likewise two sets of support structures. The server kernel support structures consist of the normal JFS structures that are pointed to by the contents of the file handle. The client kernel has the normal VFS (for example a vnode) and LFS (for example the file descriptor) support structures but instead of the normal JFS in-core inode structure the client kernel uses an rnode and maintains an rnode cache analogous to the in-core inode table. The rnode serves the same function for an NFS client as an incore-inode does to the server JFS. The security relevant contents of an rnode are described in Table 5.18.

Table 5.18. *The following fields are a subset of the fields contained in an rnode structure entry. These are derived from /usr/include/sys/rnode.h which is shipped on all systems.*

Field	Description
hash	rnode hash chain
vnode	pointer to the vnode structure
gnode	gnode structure
filehandle	information necessary to uniquely identify the file on the server.
attr	cached vnode attributes
attrtime	time the cached attributes become invalid
mtime	client time last modified
size	file size

The client kernel references files on the server via file handles. The file handle is a token provided by the server kernel to the client kernel and is given as a parameter in all NFS RPC requests. The server kernel uses the file handle to uniquely identify the object on which the requested operation will be executed. The file handle is never used or viewed by the client process. It is a kernel internal structure passed only between the client and server kernel. The client kernel does not view or manipulate the file handle and only uses it on RPC calls to the server kernel.

[4] This is the credential structure that is discussed in section 5.3.2, Process Management.

The file handle contains the device identifier, the inode number which uniquely identifies the file on disk, and a generation number which helps the server kernel resolve whether the inode referenced in the file handle has been reused. When the server kernel receives a file handle and attempts to find the associated object, one of three situations could occur. The object could already have an in-core inode in the in-core inode table (ICIT), an in-core inode may need to be fetched from disk and created, or the file could have been deleted making the file handle invalid.

The server kernel first checks to see if an in-core inode is already resident in the in-core inode table (ICIT). The kernel can determine this because, as discussed in the JFS implementation section, device number and inode number index the ICIT. If the search is successful, the server kernel VFS layer constructs a vnode for the file being accessed. If the look-up fails, the server kernel assumes there is not an in-core inode and based again on the device number and inode number, fetches the inode from disk and creates an associated in-core inode and vnode.

Invalid file handles exist when a file has been deleted and the disk inode has been released and reused. This is recognized by the server kernel through use of the generation number field contained within the file handle and also contained within the disk inode. The server kernel increases the generation number when the disk inode is reused (after a file is deleted and the disk inode is released) and on each access the server kernel compares the generation number in the inode with the generation number in the file handle. If the generation numbers do not match, it means that at some point the file had been deleted, the disk inode had been released for reuse, and the server kernel had indeed reused it. If the server kernel encounters a mismatch in generation numbers, a stale file handle error is returned to the client kernel.

A copy of the file handle is contained in the client kernel rnode and in that capacity is the functional equivalent of the disk inode contained within the in-core inode. The fields within the fileid structure are defined in Table 5.19.

Table 5.19. *The following fields are a subset of the fields contained in a file handle structure. These are derived from the /usr/include/sys/types.h file, which is shipped on all systems.*

Field	Description
inum	disk inode identifier
gen	disk inode generation number
device	device major and minor numbers

Besides the file handle, another structure called the file attributes structure is returned on NFS operations. The file attributes structure contains the basic attributes of a file and is used by the client kernel to populate the rnode and the other VFS file structures necessary to represent the remote file within the client kernel. Table 5.20 contains a description of the fields in the file attributes structure.

Table 5.20 File Attributes Structure

Field	Description
type	Type of file such as file, directory.
mode	the protection mode bits of the file
links	number of hard links to the file
uid	uid of the file owner

gid	gid of the file group
size	size of the file in bytes
used	number of bytes actually used
fsid	file system identifier for the file system
fileid	disk inode identifier
atime	last time file was accessed
mtime	last time file was modified
ctime	last time file attributes were changed

5.3.3.3.3.5 Caches

NFS has two client caches, attribute and buffer, that are used by the client kernel to conserve the number of RPC calls made to support NFS reads and writes.

The client attribute cache is advisory in nature and is only used to restrict access, not permit access. The client attribute cache is consulted to determine if the client process currently has access to the file, if it does then an RPC call is made to determine access based on the file attributes maintained by the server kernel before the operation (read or write) is allowed to proceed. The client attribute cache is updated based on the timeout period (at most 3 seconds). It is possible for a client process that actually does have access to be denied access until the client attribute cache is updated.

The client buffer cache reduces the number of requests that need to be sent to an NFS server. When a client process is reading from a file, the client **biod** issues RPC requests to the server kernel to perform read-aheads on the file and fills the client buffer cache with blocks that the client process will need in future operations. Similarly, data that is written to disk is written to the client buffer cache first and then flushed out to disk when the buffer cache fills up.

All operations in NFS result in the rechecking of access permissions. Even if the data is resident within the client kernel due to a **biod** read-ahead, an access check on the server is performed. If at any point access is denied, an error is returned to the client kernel and the server kernel sends a request to the client kernel to force the **biod** to flush the client buffer cache. It should be noted that the **biod**s are strictly a performance enhancement for NFS and are not required for NFS to work properly. See section 5.3.9.4, NFS for a further discussion of **biod**.

5.3.3.3.3.6 NFS Mount and Export Operations

An NFS server makes local file systems available to remote NFS clients with EXPORTFS. This command creates a table in the server's kernel of exported file systems as well as the *xtab* file that is used by the mountd process. mountd will return the file handle of exported file systems to requesting clients who have been granted access by the exportfs. For a description of this process see section 5.3.9.4, NFS.

NFS file systems are made available on clients via mounting. The NFS mount operation (as illustrated in figure 5.2) is a two-step process where the first step is a call by the client to determine what port the mountd is using on the server. The second step is the issuance of the MOUNT command to the remote server's mountd process with the pathname of the directory the

client wishes to mount. If the request is accepted, mountd returns the file handle of the directory to the client and the process completes. Otherwise an error is returned.

Arguments to MOUNT include the file handle, and the IP address of the server. Since every lookup operation requires a file handle pointing to the named file on the server, the mount operation seeds the lookup process by providing a file handle for the root of the mounted file system. The file handle for the root of the mounted file system is stored in the associated rnode for the mount point. The mount system call creates an entry in the client's kernel mount table, and calls the vnode interface to access the mount routine. This creates a root directory vnode for the NFS mounted file system.

From this point, all lookup operations are generated by the server using a directory file handle (starting at the root file handle) and looking up the name in that directory. It progresses down the file system hierarchy in this manner, and at each step of the way directory vnodes/rnodes are instantiated for the next step down.

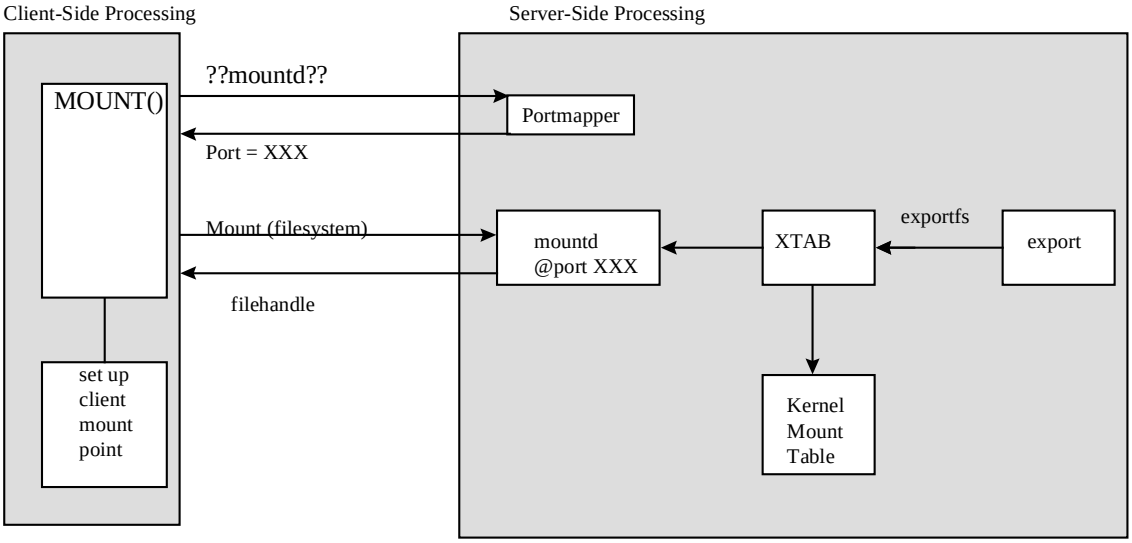

Figure 5.2: The NFS Mount Operation. *The NFS Mount Operation results in the client having access to a remote file system or access denied, depending on information in the server XTAB file.*

5.3.3.3.3.7 NFS Open Operation

The ***open*** system call at the logical file system (LFS) level generates a call to the look-up-pathname operation. This follows the normal LFS/VFS processing until it is determined that it is an NFS mounted directory. At this point the NFS client kernel processes take over the requests.

An RPC request is made to check the client process's access on the directory (to see if they have execute permissions.) The client kernel compares the client process' credentials against the client attribute cache if information is available. If the information is not available, or it is available and access was previously granted to the client process, then an RPC request is sent to the server to

check access. If the information is available, and execute access was not previously permitted for the client process, an error is returned by the client kernel to the client process for failed access.

If the above access check passes, the client kernel will begin a "lookup" for the pathname. This lookup is the same as the JFS component by component search with each component lookup resulting in two RPC requests to the NFS server, one to check for execute access and the other to return the file handle for the directory. For each component in the pathname, the server is passed the file handle from the previous component lookup (this is the directory file handle) and the string name of the component to be looked up. For each directory, execute access is checked in accordance with the DAC algorithm. The final lookup performed is with the file handle for the directory containing the object (or file) that is going to be opened. The lookup and associated access check at the server is performed with the client process credential structure that is passed in the RPC request to the server. If the object being looked up is one of the directory components, execute access is checked and if the access check fails an error is returned for failed access. If the object being looked up is the object that is being opened, the client process credential structure passed from the client kernel is used to determine if the client process has the proper access to open the file. If this access check fails, an error is returned for failed access. If the access check succeeds, a file handle is created and passed back to the client kernel. The client kernel then sends the file handle back to the server to get the file attributes (known as a getattr call) necessary to build the client kernel vnode including owner and create time. Access is again checked using the client process credential structure.

After the lookup is completed, the LFS ***open*** will result in an RPC request to get fresh attributes (getattr). If any attributes have changed since last time they were obtained, for example the create time has changed, all client caches, including the client attribute caches, are purged. If the modified time changed but the create time has not, the client data caches are purged but client attribute caches remain unless the timeout for the client attribute caches has been reached.

Once the attributes are obtained the client can build a vnode and rnode for the NFS file and places the necessary information in the client kernel system open file table and the client process's open file descriptor table and passes a file descriptor back to the client process.

5.3.3.3.3.8 NFS Read Operation

When a client process issues the ***read*** system call and the client LFS/VFS determines that this is a read of a NFS file, the NFS operations are called upon to complete the request. For each ***read***, the client attribute cache is checked prior to the RPC request to the server kernel to see if the client process had access previously and if so, the RPC call to request a read is allowed to proceed. If the client attribute cache indicates that access was not permitted an error is returned.

If the client attribute cache access check is successful an RPC request is created. This request contains the credential structure of the client process making the ***read*** request, the file handle of the file from which data is to be read, the position within the file where the read is to begin, and the number of bytes that are to be read. The server kernel checks access upon receiving the read request to see if the client process still has permissions sufficient for the read.

On a successful access check allowing read for the client process making the request, the server returns the file attributes structure that provides the file attributes on completion of the read, the number of bytes of data read, and if appropriate, an end-of-file. Presuming access checks succeed, the read data will be returned by the client biod to the client process making the request if it is already cached at the client. If a read from the server is required, the client process will be suspended and a biod will issue an RPC request to the server kernel, passing the credential structure of the client process to read the next block of the file into a buffer. The server kernel makes an access check based on the credential structure passed on the RPC and if access is permitted allows the read to continue. If the read was not permitted, a failure is returned and the server kernel also signals the client kernel to force biod to flush the cache.

5.3.3.3.4 SPECFS

The SPECFS file system is used internally by the kernel to support the device I/O subsystem and as such is not available to users directly. The SPECFS is used to support character and device special files, as well as pipes.

The RS/6000 Distributed System, like other UNIX systems, treats attached devices as files, giving each a file name abstraction in the physical file system. The device file names are associated with actual files called device special files and as described above under FSOs, these are either block or character special files. While the file system assigns an inode for a special file, the file takes no actual disk space. Special files provide handles for interfacing with devices in the form of file names that can be supplied for standard file operations for example, the open system call. The fact that a device special file is opened like any other file allows an application to perform device I/O without knowledge of the device being used.

When an application opens a device for I/O, or a pipe for IPC, the open system call causes the kernel to allocate a file descriptor, file table entry, and vnode as it would for the opening of any ordinary file. At this point however, the SPECFS, and its associated structures, are used to service the request.

5.3.3.3.4.1 SPECFS Supporting Structures

As discussed earlier in the file system implementation, each file system must have a gnode structure. For device special files there are two gnodes and two vnodes. The first gnode/vnode pair maintain the special file's place within the JFS and as described above for JFS, the vnode points to the gnode contained with the in-core inode maintained for the device special file name within the */dev* hierarchy. The second gnode/vnode pair is used to associate the device special file with a physical device (this is handled by the I/O subsystem).

There are three additional structures used by SPECFS, specnodes, devnodes, and fifonodes. The specnode contains the gnode structure and is equivalent to the in-core inode. It contains the following information:

- pointer to the next specnode (used to form linked list of specnodes)
- reference count for the specnode
- the gnode
- pointer to the gnode contained within the in-core inode for the device special file
- pointer to a devnode structure (for a device) or a fifonode structure (for a pipe)

For device special files the specnode described above points to a devnode. The devnode contains the following information:

- pointers to the next/previous devnode (this is used to form a doubly linked list of devnodes)
- device type field that holds the device major and minor number
- a reference count for this devnode
- pointer the specnode at the head of the linked list of specnodes sharing this device.

For pipes the associated special structure is the fifonode. The fifonode contains the following information:

- a size field that indicates how many bytes are currently in the pipe
- the write offset of the pipe
- the read offset of the pipe
- pointers to the buffers that hold the data as they move through the pipe

Since unnamed pipes do not have file names, the file descriptor of an unnamed pipe points to both the read and write entries in the system open file table. The system open file table, in turn, points to a single vnode which points to the gnode contained within the specnode which points to the fifonode that controls the pipe.

Because named pipes have a name within the JFS, they follow the device special file case of two vnode/gnode pairs, but instead of pointing to a devnode, the named pipe is controlled by a fifonode, the same as an unnamed pipe.

5.3.3.4 Access Revocation

For JFS and CDRFS file systems, read and write access are governed at the time the file was originally opened. In other words, access rights to a JFS or CDRFS file are revoked only when all processes that are currently accessing the file close their open file descriptors, are terminated, or the host is rebooted. On the other hand, for the NFS file system access requests at the server are validated against the current set of access rights, not the access rights in effect at the time the file was originally opened.

5.3.4 I/O Management

The I/O subsystem is responsible for sending and receiving data to and from peripheral and pseudo-devices (such as terminals, printers, disks, tape units, and network adapters).

The I/O subsystem performs the operations necessary to move data. All access checks occur at the system call level. When a user mode process attempts to perform a ***read*** on a particular device, the access check is performed when the system call is processed. If the user is allowed access, the I/O subsystem within the kernel performs the work and returns the data. The user mode process does not perform any I/O tasks except for what is available at the system call level. The kernel is responsible for picking up the request from the system call and processing it.

5.3.4.1 High Level I/O Implementation

Device drivers control and manage specific devices used by the operating system. Device drivers are installed into the kernel as kernel extensions to support a class of devices or a particular type of device. Device drivers shield the operating system from device-specific details and provide a common I/O model.

AIX supports two classes of real device drivers: character and block. AIX also supports pseudo-device drivers and adapter drivers.

5.3.4.1.1 Character Device Drivers

Character device drivers include any device that reads or writes data one character at a time. Any driver that has no associated hardware device (pseudo-device driver) is treated as a character device driver. For example, */dev/mem*, */dev/kmem*, and */dev/bus0* are character pseudo-drivers.

5.3.4.1.2 Block Device Drivers

Block device drivers support random access devices with fixed-size data blocks. Devices supported by a block device driver include hard disk drives, diskette drives, CD-ROM readers, and tape drives. Block device drivers provide two means to access a block device:

- **raw access**: the device is accessed directly, without the data to be transferred being buffered into manageable pieces by the kernel. The buffer supplied by the user program is pinned in memory, without modification, before being transferred to the device.
- **block access**: data to be sent to the device is buffered by the kernel. The kernel transmits information to the device in fixed block sizes. The buffer supplied by the user program is to be copied to, or read from a buffer in the kernel before being transferred to the device.

If a block device is accessed as raw, the device driver can copy data from the pinned buffer to the device. In this case, the size of the buffer supplied by the user must be equal to, or some multiple of, the device's block size.

5.3.4.1.3 Pseudo-Device Drivers

Pseudo-device drivers handle virtual devices, which may not have a one-to-one relationship with a physical device or no corresponding physical device. A pseudo-device driver is implemented as a kernel construct that has an I/O device-like interface.

Pseudo-devices are used wherever a set of several physical devices must be used as an integrated set. A pseudo-device provides a high-level virtual interface to aggregate devices that would otherwise require user-level code. This enables the user or application code to run without knowledge of physical device specifics.

5.3.4.1.4 Adapter Drivers

Adapter drivers are responsible for manipulating PCI cards or other system hardware and managing transfers to and from the card. Adapter drivers use AIX kernel services to handle hardware interrupts. Examples of drivers are SCSI, Ethernet and Token Ring.

5.3.4.2 Low Level I/O Implementation

5.3.4.2.1 Device Drivers

AIX device drivers are implemented as kernel extensions. Device driver routines provide support for physical devices, and are divided into two halves. The top half of the device driver runs in the kernel process environment and can be paged out, while the bottom half runs in the interrupt handler environment and is pinned. The top half of the device driver is responsible for converting requests from system calls into a format that is understood by the device, performing blocking and buffering, and managing the queue of requests of a device. The bottom half of the device driver provides the interrupt handler and routines to perform the actual I/O to the device.

5.3.4.2.2 Device Switch Table and Device Special Files

Block and character devices are accessed through the device switch table. The device switch table is stored in kernel memory and contains one element for each configured device driver. Each element is itself a table of entry point addresses with one address for each entry point provided by that device driver.

Device driver entry points are inserted in the device switch table at device driver configuration time. The driver configuration routines call various kernel services to install driver entry points into one or more entries or rows of the table. Each table entry or row in the switch table is indexed by a major number (for example, */dev/hdisk0* has major # 5 and minor # 0 device numbers). The major number assigned to a device driver is the same as the major number pointed to by the devnode for the device special file associated with the device (as described in the SPECFS section above). Block and character devices are accessed through the corresponding device special file located in the */dev* directory.

Devices are identified in the kernel through major and minor numbers. The major number uniquely identifies the relevant device driver and indexes the device switch table. The interpretation of the

minor number is entirely dependent on the particular device driver. The minor device number frequently serves as an index into a device driver-maintained array of information about each of many devices or sub-devices supported by the device driver. When an application calls ***open***, ***close***, ***read***, or ***write***, it specifies the major/minor device, and the kernel finds the rest. The device special file system figures out which entry to use based on name and major number.

Programs do not need to understand these major and minor numbers to access devices. A program accesses a device by opening the corresponding device special file located in the */dev* directory. The specnode contains a pointer to the devnode that contains the major and minor number combination specified when the device special file was created, and this relationship is kept constant until the special file is deleted. Device special files do not have to be created every time a device driver is loaded. It is only necessary to create them when a new device is initially added to the system.

5.3.4.2.3 Device I/O and The *ioctl* System Call

The ***ioctl*** system call provides an interface that allows permitted processes to control devices.

Table 5-21. Block and Character Device Special Files. *This table lists the device special files, their purpose, accessibility, and existence of* ***ioctl's****.*

Device Special File	Purpose	Accessible only by the root identity	Exclusive Use	*ioctl*'s
/dev/cd# and /dev/rcd#	Devices in the cd family refer to CD-ROM devices.	Y	N	Y, but only by root
/dev/null	The infinite sink and empty source. It is a device for discarding output or returning end-of-file on input.	N	N	N
/dev/zero	This device is the infinite NUL data source for producing NUL data.	N	N	N
/dev/fd0,/dev/rfd0,/dev/fd0.18 /dev/rfd0.18, /dev/fd0h, /dev/rfd0h, /dev/fd0.9, /dev/rfd0.9,/dev/fd01 /dev/rfd01	These devices refer to floppy disk devices.	Y	N	Y, but only by root
/dev/rmt#	Devices in the rmt family refer to the tape drive.	Y	N	Y, but only by root
/dev/gxme0	This device is the standard graphics display in the evaluated configuration.	N	Y	Y
/dev/kbd0	This device is the standard native keyboard in the evaluated configuration.	N	Y	Y
/dev/mouse0	This device is the standard native mouse in the evaluated configuration.	N	Y	Y
/dev/rcm0	This device is the Rendering Context Manager. It is used for graphics processes, such as the X Windows display. Only a single active instance may exist.	N	Y	Y

The ability to execute an ***ioctl*** on a particular device is determined by the permissions and ownership of the device special file. If a device is accessible only by the root identity, such as /dev/cd#, /dev/fd* and /dev/rmt#, untrusted users cannot attempt an ***ioctl*** because they are not able to open the device special file. To attempt an ***ioctl***, the user must open the device special file and receive a file descriptor. The file descriptor is passed with the ***ioctl*** system call.

A user who is logged in to the console of the AIX machine has ownership of the low function terminal, keyboard, mouse and graphics adapter. Each of these devices has a collection of ***ioctl*** available to modify parameters about the usage of the device. Table 5-22 provides a description of the ***ioctl*** available for each device.

Some of the ***ioctl***'s for the STREAMS devices are accessible only by the root identity. The device driver enforces this access control policy for root-only ***ioctl***'s when a user owns the device. If an untrusted user attempts to execute a root-only ***ioctl***, the call returns a permissions error.

Table 5-22. Block/Character Device Special Files and *ioctl*'s. *This table lists the device special files that contain **ioctl**'s available to the untrusted users.*

Device Special File	Description of *ioctl* Functions	Relevance
/dev/gxme0	Query parameters of the card.	The only information that is returned through the use of these ***ioctl***'s concerns physical properties of the graphics card.
/dev/kbd0	Set parameters for operation of the keyboard.	These parameters only affect the current operation of the keyboard. When the user logs out, the keyboard driver returns this device to a known state, to prevent any changes made from affecting the next user to login.
/dev/mouse0	Set parameters for operation of the mouse.	These parameters only affect the current operation of the mouse. When the user logs out, the mouse driver returns this device to a known state, to prevent any changes made from affecting the next user to login.
/dev/rcm0	Get a handle to the device, assign ownership for diagnostics, query power management status.	Ownership of the rcm0 device is assigned to a user when the user logs in. Any ***ioctl*** functions such as setting the diagnostic mode are cleared when the user logs out. When an X session is terminated, the rcm process is returned to a known state.

Table 5-23. Streams Device Special Files and *ioctl*'s. *This table lists the device special files used by streams that are accessible by untrusted users and the **ioctl**'s that can be used with them.*

Device Special File	Description of *ioctl* Functions	Relevance
/dev/ptc	Set/get attributes about the existing session, set/get values for control characters, set/get flow control, enable/disable input and output character translation, set/get window size.	In the worst case, the user can only affect their own session.
*/dev/pts/**	Set/get attributes about the existing session, set/get values for control characters, set/get flow control, enable/disable input and output character translation, set/get window size.	In the worst case, the user can only affect their own session.
/dev/lft0	Query lft information, set the default display, acquire display for exclusive use, acquire display for diagnostics.	In the worst case, the user could cause the lft to stop working in the usual fashion, thereby denying service to the users login session.

5.3.4.2.4 STREAMS

Table 5-24. Streams Device Special Files. *This table lists the device special files used by streams that are accessible by untrusted users and the capability untrusted users have to affect them.*

Device Special File	Purpose	Accessiblity
/dev/ptc	UNIX System V style PTY master interface. When this device is opened, a slave device (pts/#) is opened.	Untrusted users can open a pseudo-terminal, but only the owner of the pseudo-terminal can access the pts device after the session has begun. The act of opening /dev/ptc returns a slave pseudo-terminal that is not currently in use, so an untrusted user cannot open /dev/ptc and gain access to a session already in progress.
/dev/pts/#	UNIX System V style PTY slave interfaces. Access control is managed inside the PTY drivers so that arbitrary processes are unable to access an allocated PTY.	Untrusted users can open a pseudo-terminal using the procedure stated above for /dev/ptc, but only the owner of the pseudo-terminal can access the pts device after the session has begun. The PTY device driver enforces the access control policy for pty's. The device driver does not allow access when a user who does not own a pts attempts to open it or when a user attempts to open a pts that has not yet been allocated by /dev/ptc.
/dev/nuls	Empty data source and infinite data sink for streams operations.	Accessible to untrusted users all the time.
/dev/lft0	Graphics, keyboard, and mouse as both a traditional tty and graphical interface.	Accessible to the user logged in at the console. When the device is not in use, it is owned by the root identity with octal 600 permissions.
/dev/tty	Acts as a synonym for the /dev/pts/# device for each session.	Accessible to all users, but it points to the users /dev/pts/# entry for their session. This addresses a different /dev/pts/# entry for each session.

STREAMS are not available for general use by untrusted users outside of system-provided devices. The only STREAMS devices available to untrusted users are pseudo terminals and */dev/nuls*. These devices are outlined in table 5-24.

Configuration of a STREAMS device is dependent upon permission to open the device or modify the configuration files for the module or device. Untrusted users do not have write access to the configuration files for STREAMS modules or the ability to open STREAMS devices (except for the devices in table 5-24), so they are unable to affect the operation of the device directly or indirectly.

5.3.5 Import and Export

Each node in the distributed system can optionally have an IBM Printer 4317 and/or a 4MM SCSI tape drive. These devices serve as the import and export hardware used in the system. Refer to 5.2.6, Backup and Restore and 5.2.13, Printer Services for a description of these functions. A floppy drive is included with each system for copying data and a CDROM is included for installation of the AIX operating system and as an alternate media source for files.

The floppy, tape, and CDROM drive block and raw devices are only available to the root identity, per the direction of the RS/6000 TFM. The access control on these devices is configured during installation.

The TFM states that the administrator will not mount CDROM's that contain setuid or setgid programs.

5.3.6 Backup and Restore

Backup and restore operations are performed using the backup and restore commands, the 4MM SCSI tape drive, and tape media. The administrator can manage backup and restore from the WSM system administration utility or from the command line.

Backup and restore operations are image based, meaning that they backup and restore each individual files inodes. Because these operations are inode based, the permission bits, setuid and setgid bits, and the file's ACL (if one exists) are backed up or restored on the media or file system as a portion of the file.

Only the administrator can perform backup and restore operations. The device special file that mediates access to the tape drive is accessible by only the administrator, and the backup and restore commands are executable only by the root identity, and in the evaluated configuration have the setuid and setgid bits disabled.

5.3.7 Inter-Process Communication

The inter-process communication (IPC) mechanisms involve any method or tool that allows processes to share information. AIX provides various mechanisms, such as signals and pipes, as well as the System V IPCs (shared memory, semaphores, and message queues).

Table 5-25. Overview of IPC. *This table summarizes the IPC mechanisms provided by AIX.*

Mechanism	Characteristics	Passing of file descriptors
Unnamed Pipes	▪ allows related processes (e.g., parent/child, sibling/sibling) on same host to transfer data unidirectionally ▪ ***read()*** and ***write()*** provide one-way flow of data via corresponding file descriptors ▪ removed when all file descriptors closed ▪ since unnamed pipes have no file name or other type of handle, unrelated processes cannot participate	Unnamed pipes have no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur.
FIFOs (named pipes)	▪ similar to unnamed pipes except a named pipe has a file name created within a directory ▪ allows unrelated processes on local host to transfer data by opening the named pipe for reading or writing ▪ ***mkfifo()*** creates persistent inode of type S_IFIFO ▪ ***read()*** and ***write()*** provide one-way flow of data via file descriptors ▪ removed by ***unlink()***	Named pipes have no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur.
SysV Message Queues	▪ a process can queue a message (ENQUEUE), then an unrelated process can remove the message from the queue (DEQUEUE), using kernel memory ▪ ***msgget()*** creates message queue on local host (or returns descriptor to existing queue), based on shared key ▪ ***msgsnd()*** and ***msgrcv()*** provide access to queue ▪ IPC can be multiplexed (multiple senders/receivers); a single message queue can serve any number of processes ▪ ***msgctl()*** performs control functions, including removal ▪ queue limits defined in ***msgctl()*** man page	SysV Message Queues have no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur.
SysV Semaphores	▪ allows processes on same host to synchronize execution ▪ ***semget()*** creates semaphore on local host (or returns descriptor to existing semaphore), based on shared key ▪ ***semop()*** provides atomic manipulation of semaphore values ▪ ***semctl()*** performs control functions, including removal ▪ limits defined in ***semctl()*** man page	SysV Semaphores have no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur.
SysV Shared Memory	▪ allows processes on local host to share virtual memory, such that data stored by one process is immediately visible to another process ▪ basis for X11 shared memory socket ▪ ***shmget()*** creates a new region of virtual memory on local host (or returns descriptor to existing one), based on shared key ▪ ***shmat()*** attaches memory region; ***shmdt()*** detaches region ▪ region freed after last ***shmdt()*** ▪ ***shmctl()*** performs control functions ▪ limits defined in ***shmctl()*** man page	SysV Shared Memory has no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur.
UNIX Domain Sockets	▪ provides bi-directional IPC between processes on same host using UNIX domain protocol ▪ provides datagram socket (similar to message queue) or stream socket (similar to pipe) ▪ ***socket()*** associates socket with UNIX protocol ▪ ***socketpair()*** (available only for UNIX domain sockets)	Passing of file descriptors is blocked for Unix Domain Sockets. In the base AIX product, file descriptors are passed in a control message. In the evaluated configuration, attempts to send this control

Mechanism	Characteristics	Passing of file descriptors
	returns a pair of interconnected sockets that can be used for IPC on the same host. ▪ ***bind()*** creates a persistent inode of type S_IFSOCK and associates the socket with the pathname of the socket special file ▪ ***connect()*** requests a connection between two sockets	message are rejected within the kernel. An error is returned to the user who issued the system call and the file descriptor is not passed.
Internet Domain Sockets	▪ provides bi-directional IPC between processes on same or different hosts using Internet Protocol (IP) ▪ datagram socket (UDP) or stream socket (TCP) ▪ ***socket()*** associates socket with IP ▪ ***bind()*** associates the socket with an IP address, protocol (TCP, UDP), and port.	Internet Domain Sockets have no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur. The programming interface used for Unix Domain Sockets in the base AIX product is not present with Internet Domain Sockets.
Signals	▪ signal is a numbered event as identified in *sys/signal.h* to notify another process of a condition or situation or to take a specific action ▪ sent to a process by another process (must have same EUID unless the root identity) or by the kernel ▪ process may send signal to another process or process group using ***kill()*** ▪ kill() succeeds only if (a) UID or EUID of sender is same as UID/EUID of process to be signaled or (b) sender is the root identity ▪ signals are received asynchronously, and each process may determine how to handle individual signals	Signals have no mechanism for passing file descriptors in either the evaluated configuration or base AIX. There is no programming interface that will cause this to occur.

5.3.7.1 Unnamed Pipes

Pipes use the file system. Pipes created with the ***pipe*** system call do not have file names. This is why they are called unnamed pipes. The file descriptors of a pipe point to entries in the file table. There is one entry in the file table for each side (read and write) of the pipe. The two file table entries of a pipe point to a common vnode. After building the pipe, the ***fork*** system call is used to create a child process. A child process inherits its parent's file descriptor table. Therefore, the child inherits both sides of the pipe. Now each process (parent and child or child and child) can close whichever side of the pipe they don't intend to use since unnamed pipes can only support unidirectional data transfer. One process uses the ***read*** system call to do the "listening", while the other uses the ***write*** system call to do the "talking". The talker closes the read side of the pipe and the listener closes the write side of the pipe.

Unnamed pipes are simply file descriptors. Non-SUID programs cannot inherit a file descriptor associated with an unnamed pipe that was part of a SUID process. The inheritance rules for ordinary file descriptors apply to SUID programs as well as non-SUID programs.

All processes that share an unnamed pipe must have been related at some point in time. It is not possible for an unnamed pipe to become shared by any other means. A requirement of unnamed

pipes is that their existence must be inherited. This limits them to only related processes such as parent/child, sibling/sibling, and so forth.

5.3.7.2 Named Pipes or FIFOs

Named pipes are also known as FIFO files (first-in, first-out) and are implemented as a special file type. The internal working of named pipes is similar to that of unnamed pipes, but a named pipe has a file name created within a directory. This allows any process to open the named pipe for reading or writing. Data written to a named pipe are held in the FIFO file until another process reads from the named pipe.

Named pipes are created via the ***mkfifo*** system call specifying the appropriate permission bits.

5.3.7.3 System V IPC

System V IPCs include message queues, semaphores, and shared memory. Each of these three mechanisms has a specific application, yet their interfaces are similar. The kernel maintains three tables of descriptors, one table for each mechanism. Each of the descriptors includes a common structure used for indicating the owner and permissions for the mechanism.

The programming interface for creating, accessing, using, and controlling the System V IPC tools provides as consistent a syntax as possible between shared memory, semaphores, and message queues. The system calls ***msgget***, ***semget***, and ***shmget*** are used to create or access existing message queues, semaphore sets, and shared memory segments, respectively. The system calls ***msgctl***, ***semctl***, and ***shmctl*** are used to control message queues, semaphore sets, and shared memory segments, respectively. Controlling involves examining the IPC's current settings, changing owners, groups, and permissions, and removing the mechanism. Unique to each mechanism type are additional control options.

Some system calls vary somewhat for the System V IPC mechanisms. Messages are posted to a message queue via the ***msgsnd*** system call and retrieved from a message queue with ***msgrcv***. Semaphores are manipulated with the ***semop*** system call. Finally, the ***shmat*** system call is used for attaching a shared memory segment to a process's user space, while ***shmdt*** is used for detaching it when finished.

A process must create each System V IPC mechanism before other processes can use it. While it is in use, it is assigned to a descriptor, similar to a file descriptor, which is used as a handle when accessing the mechanism. The descriptor for each System V IPC mechanism includes a common structure (ipc_perm - which is defined in /usr/include/sys/ipc.h) that specifies the owner and group of the mechanism, as well as the permissions sets that specify read and write privileges for the IPC owner, the group, and others. The ipc_perm structure also includes the key for the IPC mechanism. A key must be specified for each message queue, semaphore set, or shared memory segment when it is created. All processes that intend to use the IPC mechanism must know the key, but this key is not part of the access control policy for the IPC objects. The access check is made at the time the object access is made. The access check is called from: ***semctl***, ***semop***,

shmat, ***shmctl***, ***msgctl***, ***msgrcv***, and ***msgsnd***. So the access check is not made at ***shmget***/***semget***/***msgget*** time, but rather at the time the object access is made.

Access permissions are checked when a user first attaches to the shared memory segment (***shmat***), every time a user performs a semaphore operation (***semop***), and every time a user performs a message operation (***msgsnd/msgrcv***).

Since access is checked at every operation, revocation is immediate for semaphores and message queues. Immediate in this case means at the time of the next request. If a request is already in progress it is not terminated (e.g. if a user is waiting on a message queue and access is revoked, that user will still get the next message).

For shared memory segments, the process specifies the desired permissions at attach time (***shmat***), and access is only granted if the user has sufficient rights (i.e., if the process requests read/write and the user has read permission only, then the request will be refused). Additional rights are not conferred to the process, even if the user gains additional rights after the initial ***shmat*** occurs. If the process requested read-only and later on the user were granted write access as well, the process would not automatically gain write access to the segment.

Finally, each message queue, semaphore set, or shared memory segment continues to exist, even if all of the processes using them have terminated, until they are explicitly removed.

5.3.7.3.1 System V Message Queues

A message queue is like a bulletin board on which a process can post a message. Another process can later pick up the message, removing the message from the queue. A message queue is created with a call to ***msgget***, which returns an identifier for the message queue. This identifier is used with subsequent system calls as a handle for the message queue.

The ***msgsnd*** system call is used to post a message to a queue and takes as a parameter the identifier of the message queue. Processes pull messages from a message queue by calling ***msgrcv***, which takes as a parameter the message queue, identifier.

The ***msgctl*** system call can be used to change owners, groups, and permissions, and to explicitly remove the IPC mechanism.

The kernel maintains a table of message queue ID data structures. One slot in this table is allocated for each message queue active on the system. The msg_perm field (as defined in */usr/include/sys/msg.h*) in this message queue ID structure contains an embedded ipc_perm structure which contains the user IDs and group IDs of the creator and owner of the message queue, along with the permissions and key value. Additional fields in the message queue ID structure keep track of timestamps and processes sending/receiving messages from the queue.

5.3.7.3.2 System V Semaphores

A semaphore is an IPC mechanism that is used to relay some condition to all participating processes. For instance, a semaphore can be used to synchronize use of some resource, such as a

file or device. Semaphores are implemented in sets, the number of which is established when the set is created. Most applications use semaphore sets that consist of a single semaphore value.

A semaphore set is created by a call to ***semget***, which returns a semaphore set identifier. This identifier is used with subsequent system calls when accessing the semaphore set. Once the semaphore set has been created, other processes can access the set by calling ***semget*** with the same key.

The ***semop*** system call is used to perform operations on semaphores (e.g., test or set/unset semaphore value).

The ***semctl*** system call can be used to change owner, group, or permissions of the semaphore set, as well as to explicitly remove the semaphore set.

The kernel maintains a table of semaphore set ID data structures. One slot in this table is allocated for each semaphore set active on the system. The sem_perm field (as defined in */usr/include/sys/sem.h*) in this semaphore set ID structure contains an embedded ipc_perm structure that contains the user IDs and group IDs of the creator and owner of the semaphore set, along with the permissions and key value.

5.3.7.3.3 System V Shared Memory

Shared memory is the quickest and easiest way for two or more processes to share large amounts of data. AIX implements shared memory by allowing processes to attach commonly defined memory regions to their own memory space. The Virtual Memory Manager (VMM) sets up and controls shared memory segments. A process creates a shared memory segment, just as a file is created. Processes attach the segment, just as processes open files. A shared memory descriptor is associated with the shared memory segment, much as a file descriptor is associated with an open file. Finally, some process is responsible for removing the shared memory segment from the system similar to a process removing a file.

To implement shared memory, a process calls ***shmget*** which returns an identifier for the shared memory instance, much like a file descriptor, to refer to the shared memory segment.

Each participating process can call ***shmat*** providing the shared memory identifier. This system call returns a pointer to the start of the shared memory segment. The participating processes can assign data to the segment or examine the data in the segment. When a process is finished using a shared memory segment, the ***shmdt*** system call should be used to detach the segment. A process automatically detaches all shared memory segments when the process terminates. Detaching a shared memory segment, even when done by the creator of the segment or the last process to have it attached, does not remove the segment from the system. A shared memory segment is removed from the system when a process with the same EUID as the creator or owner of the segment, or a process running with an effective UID of 0 (root) calls the ***shmctl*** system call to remove the shared memory segment when all participating processes have detached it. This system call is also used to change owner, group, or permissions of the shared memory segment.

The kernel maintains a table of shared memory ID data structures. One slot in this table is allocated for each shared memory segment active on the system. The shm_perm field (as defined in */usr/include/sys/shm.h*) in this shared memory ID structure contains an embedded ipc_perm structure which contains the user IDs and group IDs of the creator and owner of the shared memory segment, along with the permissions and key value. Additional fields in the shared memory ID structure keep track of timestamps and a link to the Virtual Memory Manager.

5.3.7.4 Sockets

Sockets are communication mechanisms. There are three different types of sockets: stream, datagram, and raw.

Stream sockets provide sequenced, two-way transmission of data. Datagram sockets provide connectionless messages of fixed length. Raw sockets provide access to internal network protocols and interfaces. Raw sockets are available only to the root identity and the kernel enforces creation. When running in the evaluated configuration, the system will not pass a file descriptor via a UNIX domain socket.

5.3.7.4.1 UNIX Domain Sockets

UNIX domain sockets provide stream or datagram communications between two processes on the same host computer. Server sockets must bind to and provide a known address for clients.

UNIX domain socket addresses are pathnames in the file system. The action of a server binding a UNIX domain socket creates a UNIX domain socket special file. Other processes may access a bound socket by specifying the object's pathname in a connect request.

UNIX domain socket special files are treated identically to any other file in the AIX file system from the perspective of access control, with the exception that using the ***bind*** or ***connect*** system calls requires that the calling process must have both read and write access to the socket file. The ***bind*** system call creates a persistent inode of type S_IFSOCK and associates the socket with the pathname of the socket special file.

UNIX domain sockets exist in the file system name space. The socket files have both base mode bits and extended ACL entries. Both are considered in making the DAC decision. The file system controls access to the socket based upon the caller's rights to the socket special file.

The ***socket*** system call associates a socket with the UNIX protocol. The ***socketpair*** system call (available only for UNIX domain sockets) returns a pair of interconnected sockets that can be used for IPC on the same host. The connect system call is invoked by the client process to initiate a connection on a socket. For stream sockets, ***connect*** builds a connection with the destination and returns an error if it cannot. For datagram sockets, after the client and server invoke the ***socket*** system call, there is no need for connection establishment.

5.3.7.4.2 Internet Domain Sockets

Internet domain sockets provide stream (TCP) or datagram (UDP) communications on same or different hosts using the Internet Protocol (IP). The ***socket*** system call associates a socket with IP. The ***bind*** system call associates the socket with an IP address, protocol (TCP, UDP), and port. The ***connect*** system call is invoked by the client process to initiate a connection on a socket. For stream sockets, ***connect*** builds a connection with the destination and returns an error if it cannot. For datagram sockets, after the client and server invoke the ***socket*** system call, there is no need for connection establishment. In the evaluated configuration, the ***connect*** system call for stream sockets is subject to access control on ports. See section 5.3.8.10 - TCP/IP and UDP/IP Protection Mechanisms for further information.

5.3.7.5 Signals

Signals are sent to a process by another process or by the kernel. For example, a user may issue the KILL command from the shell prompt to a background process to terminate the background process. An example of a signal sent from the kernel to a process is when a process attempts to perform an illegal act, such as a reference to an invalid memory pointer. The kernel uses a signal to terminate the offending process and, in this case, to cause a core dump of the process. Signals are also sent from the kernel to a process when driven by terminal control characters.

While signals are a form of IPC, their use is limited. They can only be implemented between processes that share a common EUID (unless the sender is running under the root identity or EUID 0). Signals cannot communicate data but simply notify the receiver of some condition; data transfer must be accomplished by some other mechanism.

5.3.8 Low-Level Network Interfaces and Communications Protocols

This section describes the protocols on which higher-level network services such as NFS and telnet rely. Some of these (TCP, UDP) are directly accessible to users through the socket interface, whereas others (IP, ARP) are used only within the TCB to support network services.

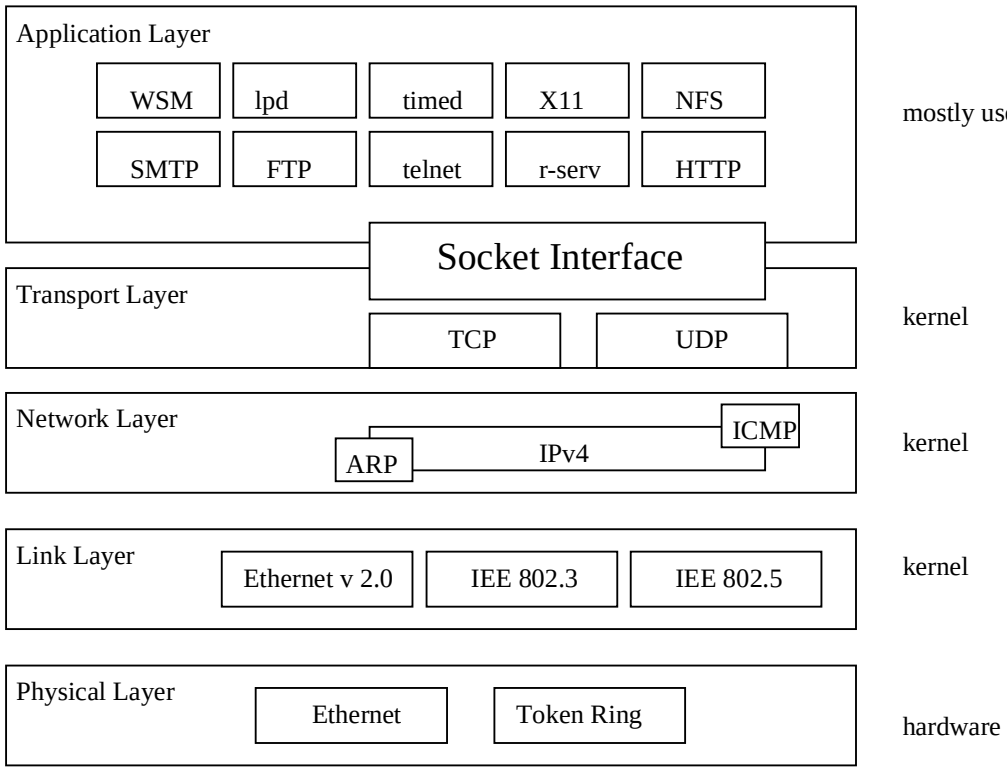

Figure 5.3: Network Protocol Overview

5.3.8.1 Network Interfaces

The network interfaces for each adapter are exported by the network interface kernel extensions. Each unique network adapter type has a network interface kernel extension. Only one instance of the kernel extension is loaded for each unique network adapter type. If a host contains multiple Ethernet adapters of the same type, only one instance of the Ethernet interface kernel extension will be loaded.

Table 5-26. Network Interfaces. *Each network interface has a kernel extension that provides the interface functions, at the interface kernel extension level.*

Interface	Description
en0, en1, en2…enN	Ethernet interfaces
tr0, tr1, tr2…trN	Token ring interfaces
lo0	Loop back interface

The device drivers for each network adapter contain transmit and receive queues that are used to temporarily buffer data before it is transmitted over the wire. These queues are internal to kernel, and are not accessible by untrusted users.

ioctl's are used with the network device drivers by the root identity to perform the following functions: enable or disable promiscuous mode, get statistics from the adapter, clear the statistics on the adapter, and enable or disable multiple addresses for one interface.

Table 5-27. Network Adapter Device Driver Interfaces. *Each device driver loaded for a network adapter has an entry in /dev to provide an interface to the hardware.*

Interface	Description
ent0, ent1, ent2…entN	Ethernet interfaces
tok0, tok1, tok2…tokN	Token ring interfaces

5.3.8.2 Internet Protocol

The Internet Protocol (IP) is a connection-less protocol that provides an unreliable packet delivery service for TCP, UDP and ICMP, and ultimately, for data to and from user processes. The service is called unreliable because packet delivery is not guaranteed and connection-less because each packet is treated independently of all others. IP includes a set of rules defining how to process packets, including when to generate error messages and when to discard packets. Part of this processing includes data fragmentation and re-assembly when required by the underlying hardware. A major function of the IP layer is to define the basic unit of data transfer used on TCP/IP networks: the IP datagram. IP also specifies the exact format of all data as it passes between hosts in the distributed system. IP is implemented within the kernel, and is not directly available to user mode software.

While AIX includes support for IPv6, only IPv4 is included in the evaluated configuration. The RS/6000 Distributed System Trusted Facility Manual provides a warning not to use IPv6. All subsequent references to IP are with respect to IPv4.

The primary functions provided by IPv4 are addressing, fragmentation and re-assembly, and routing.

5.3.8.2.1 Addressing

Each network interface has a unique 32-bit IP address. The IP addresses used on the RS/6000 Distributed System are defined in a centrally administered */etc/hosts* file. An IP address is assigned to an interface during system boot, by the IFCONFIG command called from */etc/rc.tcpip*.

5.3.8.2.2 Fragmentation and Re-assembly

IP fragments data when the datagram is too large to fit into a physical frame for the media on which the data is to be sent. Each IP datagram has an identification field that consists of a unique integer that identifies the datagram. When IP fragments data, it copies this field into the IP header of each fragment, allowing the IP module at the destination to know which fragments belong to the datagram. A datagram remains fragmented until all fragments reach the destination host, that is, they are not partially reassembled by routers. Packets may be sent with a flag to disable fragmentation, and if the medium cannot handle the packet, IP discards it and returns an error message to the sender.

5.3.8.2.3 Routing

Routing is the process of sending data packets from a source to a destination. IP performs direct routing for machines connected to the same physical network.

The ToE supports more than one physical network. IP forwarding is the mechanism used to allow different network segments of the distributed system to communicate with each other. IP forwarding happens internal to the kernel, for the sole purpose of moving packets from one network segment to another.

IP forwarding requires that a host contain at least two network adapters. Each adapter must have an IP address that represents a different network segment using a sub-net mask or a different IP network address. IP forwarding is controlled using a global kernel variable. When IP forwarding is enabled, the IP protocol examines each incoming IP packet to determine if the packet should be forwarded. If a packet is destined for a network that the host can reach on an alternate network interface, the packet is forwarded.

The RS/6000 Distributed System TFM states that no hardware routers may be used in the evaluated configuration, and that only RS/6000 systems may function as network routers, using static routes and IP forwarding.

5.3.8.3 TCP Layer

The Transmission Control Protocol is connection-oriented and provides a reliable, full-duplex byte stream for a user process. TCP uses only the services of IP when sending and receiving messages. Due to the unreliable service of the IP layer, TCP maintains its own timers, sequence numbers, and checksums to ensure the reliable delivery of data.

TCP manages:

- the establishment and termination of connections between processes
- the sequencing of data that might be received out of order
- end-to-end reliability (checksums and acknowledgments)
- flow control (preventing a sender from transmitting data faster than the destination can receive)

A socket pair uniquely identifies a TCP connection: a local IP address and port paired with a remote IP address and port. TCP ports are numbered from 0-65535 where TCP ports at 1024 and below are privileged. The RS/6000 distributed system has the ability to make any TCP ports above 1024 privileged as well. This capability is discussed in section 5.3.8.10.1, Privileged Ports. Only the root identity can bind privileged ports. The process of binding a port establishes exclusive use of that port.

5.3.8.4 UDP Layer

The User Datagram Protocol (UDP) is a datagram service that provides a connection-less, unreliable protocol for user processes. UDP provides the application layer with the same service

that IP provides, with port numbers for identifying user processes and an optional checksum to verify the contents of the datagram. UDP ports are numbered from 0-65535, where UDP ports at 1024 and below are privileged as described above.

5.3.8.5 ICMP Protocol

The Internet Control Message Protocol (ICMP) handles the error and control information between hosts on a network. ICMP reports error conditions and other information to the original source of the datagram. IP datagrams transmit ICMP messages, but TCP/IP networking software generates and processes these messages.

ICMP provides a test of the TCP/IP stacks on two hosts, and the physical hardware connections. The **PING** command generates an echo request message to a remote host. If the destination host receives the request, it responds by returning an echo reply. Successful receipt of a reply indicates that the destination host is operational, and that the IP software on the host and destination is functioning properly.

ICMP is implemented within the kernel and has only one external interface. Untrusted users can invoke the **PING** command to use echo request, but cannot generate any other ICMP message types because **PING** provides no parameter to create the different types. The **PING** command is setuid to the root identity to allow it to create the echo ICMP message.

5.3.8.6 ARP

The ARP protocol broadcasts a packet on the network that states the IP address for resolution. The host with the IP address in question responds with a reply packet stating its hardware address. This IP address to hardware address mapping is stored in the ARP cache. Not every packet operation requires an ARP lookup. The cache is consulted first.

The ARP protocol is implemented within the kernel and has no external interfaces. Untrusted users or the root identity use the **ARP** command to query the current ARP cache. The ARP cache is stored within protected kernel memory, and cannot be modified by untrusted users.

Access to the ARP cache is performed via unique ***ioctl***s performed on the ARP socket. Delete or invalidate operations are only permitted by the root identity, and are controlled by the kernel in the ***ioctl*** processing associated with the ARP entry deletion or cache invalidation. Arbitrary ARP cache entry invalidation would pose a performance problem hence the restriction to the root identity.

5.3.8.7 RPC

Remote Procedure Call (RPC) provides a mechanism for one host to make a procedure call to another machine on the network. The RPC protocol is defined on top of both TCP and UDP - but in the evaluated configuration the RPC protocol relies on the TCP/IP protocol to communicate messages between hosts.

RPC is used to communicate requests and responses between clients and servers. The client makes a request using the program name, version number, and the procedure to execute. The request also includes an authentication token, in the case of the NFS application's use of RPC it is a credential structure that is used on the server to determine access. If the requesting user has access, a reply message is returned that includes the result of the remote procedure. If the user is denied access, a message is returned specifying the reason the user was denied access.

5.3.8.8 Bind and Connect Interfaces to the Stack

The user interfaces to the TCP/IP stack exist as system calls. The path to setting up a socket based communication involves the ***socket***, ***bind***, ***listen***, and ***connect*** system calls. A program issues the ***socket*** call and receives a descriptor. The ***bind*** system call is then issued to bind the socket to a port. The privileged port mechanism is invoked at this time, as described in section 5.3.8.10.1, Privileged Ports. If the ***bind*** is successful, a ***listen*** system call is performed, and the program waits for a connection. A ***connect*** system call is issued from another process on the local host or from a host on the network, which sets up the connection for use. At this point for TCP connections, the access control on ports is invoked to allow the connection to be made or denied. See section 5.3.8.10.2, Access Control on Ports for further details. Until the socket is shutdown, the processes can use the send and receive system calls to exchange data.

5.3.8.9 TCP/IP Stack

Table 5-28. User Interfaces to the TCP/IP Stack. *An untrusted user invokes these system calls to facilitate network communications using TCP or UDP.*

System Call	Description
accept	Accepts a connection on a socket to create a new socket.
bind	Binds a name to a socket.
connect	Connects two sockets.
getdomainname	Gets the name of the current domain.
gethostid	Gets the unique identifier of the current host.
gethostname	Gets the unique name of the current host.
getpeername	Gets the name of the peer socket.
getsockname	Gets the socket name.
getsockopt	Gets options on sockets.
listen	Listens for socket connections and limits the backlog of incoming connections.
recv	Receives messages from connected sockets.
recvfrom	Receives messages from sockets.
recvmsg	Receives a message from a socket.
send	Sends messages from a connected socket.
sendmsg	Sends a message from a socket by using a message structure.
sendto	Sends messages through a socket.
setsockopt	Sets socket options.
shutdown	Shuts down all socket send and receive operations for that socket.
socket	Creates an end point for communication and returns a descriptor.
socketpair	Creates a pair of connected sockets.

The TCP/IP stack is implemented as a kernel extension and exports intra-TCB interfaces to the network device drivers which in turn facilitate the transfer of data to the network media. All of the network protocols (TCP, UDP, IP, ARP, and ICMP) are implemented within the kernel extension. The only external interfaces to the TCP/IP stack are the system calls listed in table 5-28, the ARP command and the PING command.

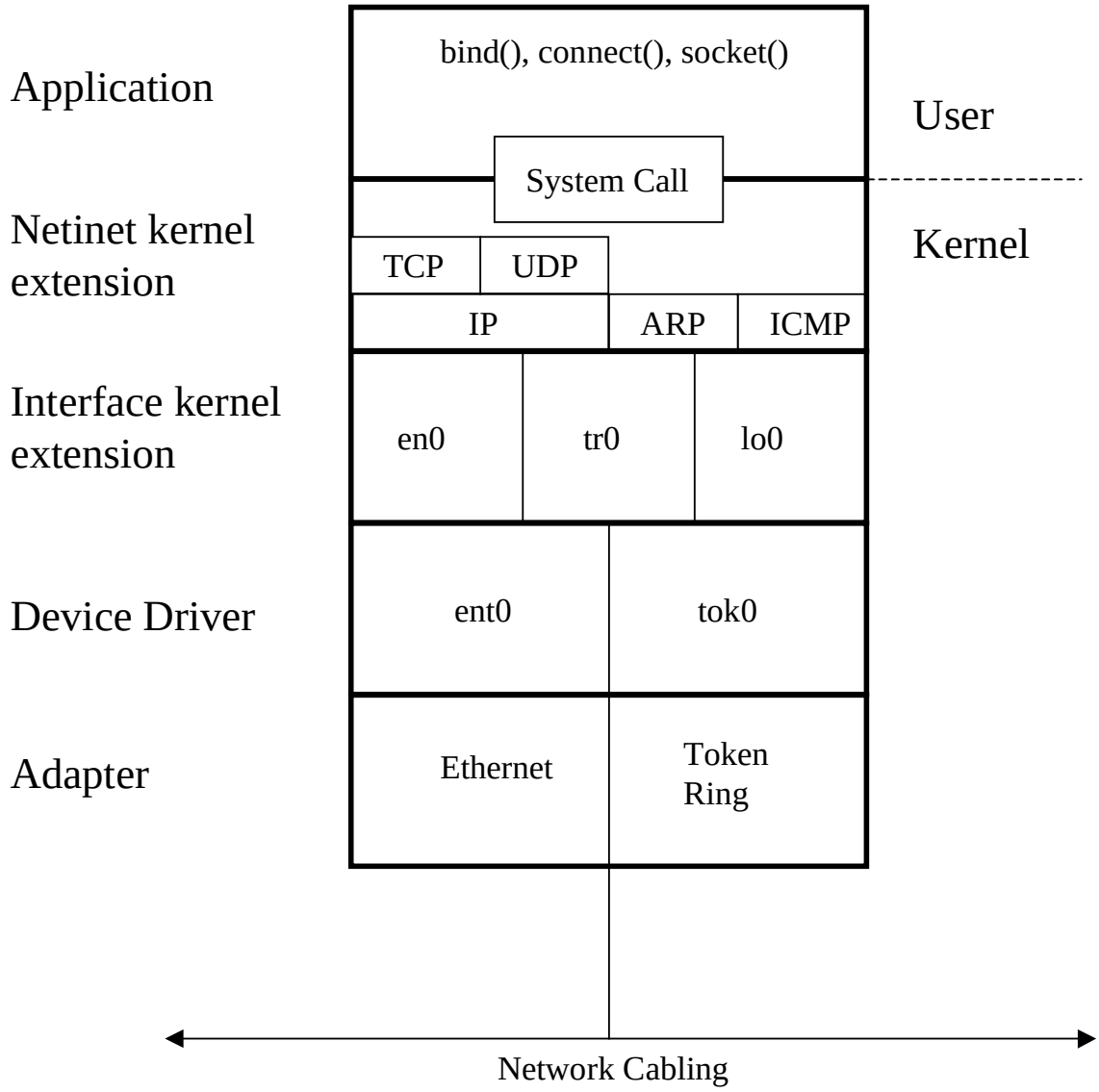

Figure 5.4: AIX TCP/IP Stack

The loop back interface is contained at the interface kernel extension layer. All kernel functions and security checks apply to the loop back interface.

A user mode process executes the ***send*** system call to transmit data. The ***send*** system call requires a file descriptor and a buffer containing the data to be sent. When the ***send*** system call is invoked, the user's buffer is transferred from user memory to kernel memory. This copy operation happens only once. Subsequent access to the buffer is made using a pointer. The kernel leaves enough space free at the beginning of the buffer to insert the protocol and IP header.

The path that is followed within the kernel is dependent on the type of socket in use. When an Internet socket is used with the TCP protocol, the receiving socket is checked to make sure it has enough space in it's buffer allocated to receive the data and a TCP header is created in the buffer on the local system. The next step is to fill in the IP header for the packet and determine the correct route. The correct route is discovered by executing a function to determine the interface and source address for the packet. The packet is then transferred to the interface kernel extension, which formats the packet for the correct network type and passes it to the device driver. The final step is for the device driver to transfer the packet to the network adapter using a DMA operation.

When a packet is received on a host, it travels through the adapter, device driver, and interface kernel extension as described above, and is processed by the kernel before being sent to the user mode process, as a buffer.

5.3.8.10 TCP/IP and UDP/IP Protection Mechanisms

TCP and UDP port privilege and access control mechanisms protect the hosts in the distributed system. Privileged ports block anyone but the root identity from binding to a port that has been designated as privileged. The access control mechanism performs a check to determine if a requesting user is allowed access to a specified port.

5.3.8.10.1 Privileged Ports

Ports below 1024 are automatically privileged. This privilege check is implemented in the TCP/IP stack. When an untrusted user attempts to bind a socket to a privileged port, the bind is not completed and an error is returned. A function exists in the kernel that determines if a port is privileged during a bind attempt.

The evaluated configuration has an additional privileged port mechanism. This mechanism allows the root identity to specify TCP ports above 1024 as privileged.

The */etc/security/priv* file contains the definition of privileged TCP ports above 1024. During system boot, the */etc/security/priv* file is read, and a kernel array is populated with the privileged ports. When a user attempts to bind to a port, the kernel checks that the port is not below 1024 and that it is not included in the kernel array containing the list of privileged ports above 1024. If the kernel determines a port is privileged, the bind is not completed and an error is returned. If the port not privileged, the user is allowed to bind a socket to it.

Ports 6000-6255 are reserved for X Windows usage, and are configured to be privileged using the mechanism above. NFS (2049) and WSM (9090) are also configured as privileged ports, using the same mechanism.

5.3.8.10.2 Access Control on Ports

The only path a packet may follow once it has been received on the host is through the access control mechanism, which is implemented in the TCP/IP stack.

During system boot, two steps are performed to initialize access control on ports. First, the */etc/security/acl* file is read, and the UID, GID, and host information for the access check on a port are written into a kernel structure. Second, the */etc/security/services* file is read, and an ACL is created allowing all access on the port for the specified service. Telnet, ftp and rexec are listed in the */etc/security/services* file for an ACL allowing all access. This presents no security issue because use of telnet, ftp, and rexec requires that the user requesting service provide a valid username and password before access is granted. The HTTP port is also included in the */etc/security/services*. HTTP is a public read-only service.

The ACLs contain UIDs and GIDs that are permitted to establish connections on the specified TCP Internet Domain ports on the host. The ACL is of the form service/port, host/subnet, and user/group. The host or subnet specification can be empty, in which case the user or group refers to connections originating from any host in the system.

The ACL check is performed based on the effective user ID that is responsible for opening the connection. The login and effective UIDs are passed from the originating host to the destination host in an authentication header. The login UID is utilized for auditing operations. The authentication header is implemented using a TCP option.

When a host attempts to connect, TCP uses a three-way handshake to setup the communication path. This handshake involves sending a SYN packet, receiving an ACK (acknowledge) from the remote host, and sending an ACK to tell the remote host that the connection is ready for use. The kernel on the local host appends the authentication information to the SYN packet as a TCP option. The kernel on the remote host performs the access check based on the values contained in the TCP option and the access control information contained in kernel memory. If the UID of the sender is allowed to connect to the port, an ACK is returned. If the UID is not allowed to connect to the port, an RST (reset) is returned, and the connection is dropped on the remote side.

The RS/6000 Trusted Facility Manual explains that the ports used by NFS, X11, **SENDMAIL**, and WSM must be configured with an ACL that permits only the root identity to connect.

- For NFS, this is used to prevent a user on a client from forging requests to an NFS server that would permit the user to become the root identity.
- For X11, this is used to protect the administrator if he/she chooses to use Internet Domain Sockets.
- For **SENDMAIL**, this is used to prevent users from forging e-mail.
- For WSM, this is used to prevent non-administrative users from directly making requests to the WSM server.

5.3.8.11 Address Mapping

Address mapping is the process of translating a name or Internet address to a hardware address. There are two types of address mapping: Internet address to physical address and host name to Internet address.

5.3.8.11.1 Internet to Physical Address Mapping

Each network interface card installed in a host in the distributed system has a hardware address. This hardware address cannot be changed, and is branded on the card at the time of manufacture. This physical address is translated to an IP address via the ARP protocol.

5.3.8.11.2 Host Name to Internet Address Mapping

Each network interface card installed in a host is also assigned an IP address. When a user attempts to connect to another host using a host name, the */etc/hosts* file is referenced to provide the IP address to hostname mapping.

Domain name service (DNS) is not included in the evaluated configuration.

5.3.9 Network Applications

This section summarizes the various application protocols included in the distributed system, as well as inetd, which receives requests for network services and initiates server daemons.

Table 5-29. Network Applications. *The following applications are used in the distributed system to facilitate the exchange of information.*

Protocol	Description	Security Features	See Also
telnet	a remote terminal session on a host, from another host in the distributed system	Username and password must be supplied during the initialization of the session for I & A and user can change identity if password is known	5.3.9.2 telnet and 5.3.11.2.1 Login with telnet
file transfer protocol (FTP)	file transfer between hosts in the distributed system	Username and password must be supplied during the initialization of the session for I & A and user can change identity if password is known	5.3.9.3 FTP and 5.2.11.2.7
simple mail transfer protocol (SMTP)	the exchange of electronic messages between hosts in the distributed system	privileged TCP port 25, allowing only connections from the root identity on any other host to this port	5.3.9.5 SMTP and 5.3.14 Mail
WSM	systems management interface	privileged TCP port 9090, allowing only connections from the root identity on any other host to this port	5.3.9.6 WSM and 5.3.11.2.6 WSM login
BSD r-services	remote login, copy, shell, and job execution	authenticated r-services (rexec) require providing an identity and password for I & A and user can change identity if password is known unauthenticated r-services (rcp, rsh, rlogin) force the user identity (LUID) to be the same and user is not permitted to switch identity using the ***-l*** option	5.3.9.7 rlogin and 5.3.11.2.2, Login with rlogin 5.3.9.8 rsh and rcp, 5.3.11.2.3-4, Command execution with rsh-rcp , 5.3.9.9 rexec and 5.3.11.2.5, Command execution with rexec
network file system (NFS)	file sharing based on exported file systems	Each RPC request contains user authentication information. NFS server	5.3.9.4, NFS

Protocol	Description	Security Features	See Also
		performs DAC based on user identity. Requests permitted only from kernel (privileged TCP port 2049). Server and client both run internal to the TCB.	
hyper text transfer protocol (HTTP)	information exchange between clients and servers	Server provides a read-only service with access restricted to AIX information library documents only, e.g. TFM, SFUG.	5.3.9.10 HTTP
line printer daemon (LPD)	remote printing capability	privileged TCP port 515, allowing only communications from the root identity on any other host to this port. Client and server are setuid the root identity.	5.3.13 Printer Services
X Windows	graphical operating environment	privileged TCP ports 6000+x, where x is the display number being used (x: 0 - 255).	5.3.9.13 X Windows
timed	network based time synchronization	None	5.3.9.14 timed

Table 5-30. Port Protection. *The following table summarizes the server/client port protections in the evaluated configuration.*

Service	Protocol	Server Port	Privileged Port (server)	Root Access Control (client)
telnet	telnet	23 TCP	Y	Y[5]
ftp	ftp	21 TCP	Y	Y[6]
www	http	80 TCP	Y	N[6]
mail	smtp	25 TCP	Y	Y[6]
rlogin	rlogin[7]	513 TCP	Y	Y
rsh	rsh[7]	514 TCP	Y	Y
rcp	rsh[7]	514 TCP	Y	Y
rexec	rexec	512 TCP	Y	Y
lpd	lpd	515 TCP	Y	Y
wsm	wsm	9090 TCP	Y	Y
nfsd	nfs	2049 TCP	Y	Y
mountd	mount	Assigned[8]	Y	Y
statd	stat	Assigned[8]	Y	Y
lockd	lock	Assigned[8]	Y	Y
portmap	sunrpc	111 TCP	Y	Y
timed	time	525 UDP	Y	N

The following subsections summarize the client and server daemon interactions in the distributed system with respect to process privilege and identification/authentication of user if applicable. The TCB protection mechanisms which exist will be used to provide protection for the TCP/IP

[5] Differs from common practice, most UNIX systems allow unprivileged clients.

[6] Usable only for access to public information (TFM, SFUG).

[7] Protocol extended to pass LUID.

[8] Assigned by portmap, protection set up during initialization.

connection using the privileged ports and access control on ports described in section 5.3.8.10, TCP/IP and UDP/IP Protection Mechanisms.

5.3.9.1 inetd

inetd is a daemon that listens for connection requests for TCP or UDP based services, and starts the appropriate daemon for the requested service. When inetd starts, it reads from the */etc/inetd.conf* file. *inetd.conf* provides a list of services (ftpd, rlogind, telnetd, rshd, rexecd, and optionally WSM if remote enabled) that are initiated by inetd. inetd runs with the root identity, and is executed by ***rc.tcpip*** during system initialization.

The interface to inetd consists of a number of Internet domain ports, both TCP and UDP, being listened to. When inetd detects that a connection has been made to one of those ports, taking into consideration the port protection mechanism, the command which is defined in */etc/inetd.conf* for that port is started and passed the new connection. That is the extent of the interface. The user does not have any other mechanism for accessing inetd.

5.3.9.2 telnet

The telnet protocol provides a remote terminal session from another host in the distributed system. The telnet client is a privileged program run by the user initiating the telnet session. The user must supply a valid user name and password to initiate a session on another host. When a client initiates a connection to TCP port 23 on a server, inetd spawns an instance of the telnetd server. The telnetd server invokes LOGIN to perform identification and authentication. A user can change to an alternate identity if a proper identity and password are provided. Audit on the host server is based on this identity.

The telnet client runs setuid to the root identity to create a connection to TCP port 23 on the remote host. TCP port 23 is protected by an access control that disallows connections from any other user-id but 0. The telnetd server runs with a UID of 0.

The server running on the host creates a shell that runs with the identity of the user that logged in to the server.

5.3.9.3 FTP

The FTP protocol is used to create an interactive session for performing file transfers between hosts. When a client initiates a connection to TCP port 21 on a server, inetd spawns an instance of the ftpd server. The FTP client prompts for the user's name and password and transmits this information to the ftpd server.

The ftpd server invokes the authenticate subroutine to perform I&A. The authenticate subroutine is described in section 5.3.10.2, Common Authentication Mechanism. If the authentication procedure is successful, the effective user-ID of the ftpd is set to the user-ID that logged in. The ftpd server on the host issues system calls on behalf of the logged in user, using the identity of the logged in user. Audit for the FTP session is determined by the audit parameters for the user account.

FTP Client software runs setuid to the root identity to create a connection to TCP port 21 on the remote host. TCP port 21 is protected by a port-level access control that disallows connections from any other user-id but 0. The ftpd server runs as UID 0 until it performs a setuid to the UID of the user whom authenticated to the server.

Anonymous FTP is not supported in the evaluated configuration. There is no FTP or anonymous account defined in the */etc/security/passwd* file.

Table 5-31. Relevant FTP Protocol Commands. *The following commands are the security relevant commands used by the FTP protocol*

Command	Description	Relevance
USER	a string identifying the user	Used in I&A
PASS	a string identifying the users password	Used in I&A
CWD,XCWD	change working directory	User must have execute access on the directory to enter it
RETR	retrieve a file	User must have access to the file to download it
STOR	store a file	User must have access to the directory to upload the file, and a file must not exist with the same name, but belonging to another user
APPE	append or create a file	User must have access to the file or the file must not exist
RNFR	rename from filename	User must have write access to the parent directory and access to the file to be renamed
RNTO	rename to filename	User must have write access to the parent directory and access to the file to be renamed
DELE	delete named file	User must have ownership of the file to delete
RMD,XRMD	remove directory	User must have ownership of the directory to delete it
MKD,XMKD	make a directory	User must have rights in the directory to create a new directory
SITE,CHMOD	changes the permission bits for a specified file	User must be the owner of the file to adjust permission bits

5.3.9.4 NFS

NFS is an RPC based protocol, with a client-server relationship between the host having the file system to be distributed and the host wanting access to that file system. NFS server daemons called nfsd daemons run on the server and accept RPC calls from clients. NFS servers also run the rpc.mountd daemon to handle file system mount requests and pathname translation. On an NFS client, the biod daemon runs to improve NFS performance.

On the client host, each process using NFS files is a client process. The client process issues system calls to access NFS mounted files, and the client kernel services these system calls by issuing RPC calls to the NFS server from which these files were mounted. The virtual file system layer extends the various operations (such as read/write) to work with remotely mounted file systems. For example, on an NFS client, a user process executes the ***chmod*** system call on an NFS-mounted file. The VFS layer passes the system call off to the NFS specific set of operations in the client kernel which then executes an RPC to the NFS server which sets the permissions on the file as specified in the client process's system call. The NFS server replies to the client kernel

whether or not the call succeeded and on success returns a new file attribute structure. The client kernel in turn returns a success or failure to the client process that issued the system call.

5.3.9.4.1 RPC Components

There are only two daemons that are included in NFS proper: the server nfsd; and the client biod. However, there are several RPC components that directly support NFS, namely the portmap daemon and the rpc.mountd, rpc.statd and rpc.lockd daemons. In the evaluated configuration NFS is the only trusted service using the RPC protocol and so the discussion of these RPC components is provided here.

The rpc.mountd, rpc.statd and rpc.lockd daemons are all protected by access control entries on their dynamically assigned ports. During system initialization, the DACINETADM command is issued twice. The first time initializes the privileged port mechanism for ports above 1024 and any access-control entries for ports which are defined in the */etc/security/acl file*. The second time DACINETADM is executed it querys the portmap daemon to discover the dynamically assigned TCP ports and sets up the UID 0 access control entries for those ports.

5.3.9.4.1.1 portmap

RPC servers are started during the boot process and run as long as the host is up and running. Instead of using pre-assigned ports and a super-server such as inetd, RPC servers are designated by service number. The file */etc/rpc* contains a list of RPC servers and their program numbers. RPC services still use TCP port numbers to fit the underlying protocols, so the mapping of RPC program numbers to port numbers is handled by the portmap daemon.

When an RPC server begins execution, it registers the port number it is listening on and the RPC program number and version it serves. An RPC client kernel must contact the portmap daemon on the server kernel to determine the port number used by an RPC server prior to attempting to send RPC requests to that server. The portmap daemon returns the port number of the RPC server to the client kernel to allow subsequent requests to be made.

In the evaluated configuration, NFS is the only trusted service that uses RPC. Untrusted software may register with the RPC server to advertise untrusted RPC services. Untrusted software may not override system level RPC services that have been allocated during system boot. The kernel must free up RPC program and port numbers before they can be reused.

If the portmap daemon dies, clients will be unable to locate RPC daemon services on the server. A server without a running portmap stops serving RPC-based applications such as NFS.

5.3.9.4.1.2 Mounting Network File Systems and rpc.mountd

NFS services require the rpc.mountd daemon be present to be able to service requests. On the server, the */etc/exports* file determines which file systems the NFS server will allow clients to mount via NFS. At system initialization, the kernel checks to see if the */etc/exports* file exists and, if it does, runs EXPORTFS to make the file systems available for clients by placing them in the server kernel export file system table. An administrator may cause other file systems to be exported once

the server is up and running. The */etc/exports* file and the server kernel exported file system table may not match exactly.

The */etc/exports* file contains the list of exported file systems and also contains any restrictions or export options placed on each. The following table indicates the options that may be placed on file systems within the exports file. If no restrictions or options are indicated in the exports file, any client can mount the file system in any mode (e.g. read and write).

Table 5.32: NFS Export Options

Option	Function
RW=host:host	Limit hosts that can write to the file system. If no hostname is given, any NFS client may write to the file system.
RO	Prevent any host from writing to the file system. This is checked on operations, not on mounts—i.e. the file system could be mounted read/write, but on any write operations a failure error will be returned.
Access=host:host	Restrict access to only the named hosts in this list.

5.3.9.4.1.3 NFS File Locking

NFS file locking is performed using the network lock manager rpc.lockd and the network status monitor rpc.statd. These daemons run on both the client and the server and the client-side daemons coordinate file locking on the NFS server through their server-side counterparts.

rpc.lockd processes lock requests generated locally or remotely by another lock daemon. If a local client on a remote machine requests a lock, the rpc.lockd daemon forwards the lock request to the remote machine.

rpc.statd works in conjunction with rpc.lockd, providing crash and recovery features for the locking services of NFS.

5.3.9.4.2 NFS Server Daemon: nfsd

The nfsd daemon runs on the server and services client requests for NFS data. nfsd is started only if the */etc/exports* file indicates that there are file systems capable of being NFS mounted on clients. The nfsd is started in user space, makes a system call into the kernel and never returns. The system call executes the NFS code in the server kernel. Since the nfsd has not been reworked to take advantage of the multi-threading capabilities of the AIX kernel, multiple versions of the nfsd are started. Each version of the daemon handles one request at a time.

An ACL on the NFS service port (TCP 2049) permits only connections that originate from the root identity. TCP port 2049 is a privileged port, so unprivileged processes cannot bind to the port and advertise as NFS.

5.3.9.4.3 NFS Client Block I/O Daemon: biod

The biod daemons perform block I/O operations for NFS clients, performing some simple read-ahead and write-behind optimization. There are multiple instances of biod so that each client process can have multiple NFS requests outstanding at any time. It should be noted that the biod's

are only for improved performance on NFS read and write requests. They are not required for the correct operation of NFS.

The biod daemon runs on all NFS client systems and issues requests to NFS servers for information, on the part of the user. The biod daemon is invoked at system boot time, and runs continuously. The client-side biod daemon is the part of NFS that interacts with the buffer cache. The file buffer cache mechanism is used to provide throughput, similar to that achieved with a JFS file pre-fetch operation that reduces the number of disk accesses when a process is reading a file, by moving file blocks between the server kernel and client kernel.

The client-side caching mechanisms (buffer caching and file attribute) reduce the number of requests that need to be sent to an NFS server. When a process is reading from a file, the client kernel performs read-ahead on the file and fills the buffer cache with blocks that the process will need in future operations. Similarly, data that is written to disk is written into the cache first and then flushed out to disk when the cache fills up.

When the client process requests a read file operation, the client kernel makes a request to the biod daemon which sends an RPC request to the nfsd on the server to read additional data as a pre-fetch activity.

5.3.9.5 SMTP

Simple Mail Transfer Protocol (SMTP) is used to facilitate the exchange of electronic messages between users in the distributed system via a TCP/IP connection.

The SMTP client (**SENDMAIL**) runs setuid to the root identity to create a connection to TCP port 25. TCP port 25 is protected by an access control that disallows connections from any other user-id but 0. The execution of **SENDMAIL** as a system daemon runs with UID 0. The **SENDMAIL** application is both the client and server in the evaluated configuration. **SENDMAIL** is discussed in section 5.3.14, Mail.

5.3.9.6 WSM

The Web-based System Manager (WSM) is a client-server administrative interface for the distributed system. WSM is the only interface allowed for system administration while the system is operating in multi-user, secure state. The TFM states that the administrator should not use SMITTY, the text based management interface, when the system is operating in secure state, and should only use WSM.

WSM allows the administrator to perform system administration tasks involving users, software, devices, printers, logical volumes, file systems, backup/restore, network settings, processes, and subsystems. WSM acts as a front-end shell for the execution of commands by providing dialog boxes to assist in formatting command parameters.

The commands called from the GUI are the same commands used to administer the system from the command line. The ability to execute a command is enforced at the command interface, by the

permissions on the command. WSM provides the same functionality as the command line, without the administrator memorizing the exact parameters for each command.

The WSM client runs as an **X** client, and allows the administrator to administer the local machine or a remote machine. The administrator is required to authenticate locally or remotely, as outlined below.

The WSM server is a daemon that is invoked by inetd following a connection to TCP port 9090. TCP port 9090 is protected by an access control that disallows connections from any other user-id but the root identity.

5.3.9.6.1 Local Mode

The WSM client provides the capability to administer the local machine in local mode and runs with the security attributes of the user who executed it. Untrusted users can use WSM in local mode.

The user who executes the WSM client locally can perform a "Switch User" to use the WSM client as a different user or as the root identity. To successfully "Switch User", the correct password for the account being switched to must be provided.

The WSM client does not communicate with a WSM server when operating in local mode. The WSM client process performs a ***fork*** to create a new process and ***exec*** to execute the requested command directly. The new process is created using the effective UID of the WSM client process, and runs with the security attributes of the currently authenticated user.

If an administrative user wishes to perform administrative functions on the local machine they invoke the "Switch User" option from the GUI. The "Switch User" option presents a dialog box for the administrator to enter the username root and the root identity's password. "Switch User" then calls the common authentication mechanism and if the authentication information is correct, the effective UID of the WSM client process is changed to zero, causing all subsequent commands to be executed with an effective UID of zero.

The "Switch User" is not limited to the root identity. It is limited by knowledge of the password for the account you wish to "Switch User" to and the current state of login restrictions against the account.

5.3.9.6.2 Remote Mode

In remote mode, the administrator is interacting with the WSM server on the remote machine, while executing the WSM client on the local machine. Only the administrator can use WSM in remote mode.

To use WSM in remote mode, the administrator executes the same client as local mode. The command to launch the WSM client contains a parameter that specifies which host in the distributed system to connect to. This parameter causes the communication between the client and server to take place using a TCP socket.

To administer a remote machine, the administrator must complete the following steps:

1. Login to the local machine with a non-root user ID.
2. Start X Windows.
3. Su to the root identity. This allows the administrator to connect to TCP port 9090 on the remote host.
4. Start WSM Client in remote mode, specifying the remote machine to administer using the command line parameter.
5. Login to the WSM server by providing a non-root user ID and password.
6. Perform the Switch User menu option from the WSM Client to assume the root identity on the remote machine.

The common authentication mechanism is called to process the login request to the WSM server. It the user successfully authenticates, their UID, GID and login UID are stored. The login name on the remote side must be their non-administrative user-id. To assume the root identity, they must perform a second "Switch User". If the "Switch User" to the root identity is successful, the WSM server stores zero for the user ID.

The login UID is not modified by the Switch User menu option. If the system generates an audit record as a result of a command that is executed by the user on the remote side, the audit record on the remote machine contains the correct login UID. If they have successfully performed a Switch User to the root identity, their login UID is not changed, so an audit record will contain the login UID that was set on their initial login to the WSM server.

The WSM Server executes as UID 0. When the currently logged in user requests a command, the WSM server process performs a ***fork***, a ***setgid*** to set the group ID for the child process, a setuid to change the UID of the process from zero to the UID for the logged in user of the WSM server and an ***exec*** to execute the command. The login UID value for the child process is set by the WSM server using a system call before it performs the setuid, ensuring that any audit records that are cut on the remote machine due to command execution will contain the correct value for login UID.

5.3.9.7 rlogin

This protocol allows a user to login to a remote host. The server running on host creates a shell that runs with the user's identity (same as on client). DAC and audit are performed using this identity.

The rlogin client passes the user's identity to the rlogind server. When a client initiates a connection on TCP port 513 on a server, inetd spawns an instance of the rlogind server.

The user is not permitted to switch identity using the ***-l*** option. The client runs setuid to the root identity and utilize TCP port 513. TCP port 513 is protected by an access control that disallows connections from any other user-id but 0.

5.3.9.8 rsh, rcp

rsh provides a shell on a remote host. The server running on host creates a shell that runs with the user's identity (same as on client). DAC and audit are performed using this identity. The RCP client software connects to the rshd server, but is separate from the rsh client software.

The rsh client passes the user's identity to the rshd server. When a client initiates a connection on TCP port 514 on a server, inetd spawns an instance of the rshd server. The user is not permitted to switch identity using ***-l*** option. The client runs setuid to the root identity and utilizes TCP port 514. TCP port 514 is protected by an access control that disallows connections from any other user-id but 0.

5.3.9.9 rexec

This protocol provides the capability to execute jobs on a remote host.

The user must supply a valid user name and password to initiate an rexec session. When a client initiates a connection on TCP port 512 on a server, inetd spawns an instance of the rexecd server. The rexecd server invokes the common authentication mechanism directly to perform the I & A. A user can change to an alternate identity if a valid user name and password are provided. Audit on the server host is based on this identity.

The client and server run as setuid root and utilize privileged TCP port 512. TCP port 512 is protected by an access control that disallows connections from any other user-id but the root identity. Client and server must run as setuid root to connect or bind to that port.

The server running on host executes command with identity of the user that logged in to the server.

5.3.9.10 HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol. HTTP is used within the RS/6000 Distributed System to provide administrators and users an interface with the Lotus Domino Go Webserver. The Webserver provides an interface to users and administrators to view and search public information such as the TFM, SFUG, and other system reference information through cgi scripts.

The HTTP protocol has no access control defined on the connect side because information provided is public (system documentation). The TFM tells the administrator that the HTTP protocol and the web server are only to be used for presentation of public data. The interface provided by HTTP consists of messages, requests, responses, and entity as described below.

WSM does not use the web browser or Webserver. Instead it executes as a standalone application.

5.3.9.10.1 Messages

HTTP messages consist of requests from client to server and responses from server to client offering an interface to the administrator and to untrusted users. These only concern communication options such as the date and time at which the message originated, or any type of transfer encoding that has been applied to the message body. This interface is not security relevant and it only affects the ability of the client and server to communicate.

5.3.9.10.2 Requests

Table 5.33 HTTP Request Methods. *The HTTP request interface provides a number of methods by which to request services from the web server.*

Method	Description	Security Analysis
OPTIONS	The OPTIONS method is a request for information about the available communication options.	This interface is not security relevant. It only affects the ability of the client and server to communicate
GET	The GET method retrieves the identified entity. If the identified entity refers to a data-producing process the data produced is returned as the entity.	The HTTP configuration file (*/etc/http.conf*) specifies the root of the file system available over the Webserver. Users are restricted to performing GET operations on the files within the specified directory. (Attempts to "back out" of the specified directory using "../" are thwarted.) Requests for files over the HTTP interface are relative to the specified directory to the requested file name. In the evaluated system all files in the specified directory intended for public access and are world readable.
HEAD	The HEAD method is identical to GET except only the meta-information is returned.	This interface allows only meta-information to be returned to the user. It is no worse than what can happen with the GET interface. The analysis performed for the GET interface applies here as well.
POST	The POST method is a request to the server to accept the enclosed entity. POST is a method to provide a block of data, such as the result of submitting a form, to a data-handling process.	In the evaluated system the POST method is used as the interface to the document search utility (docsearch). The user may use the provided form which converts form input into POST requests specifying a search string and selecting documents to search. Alternatively, the user may directly interface with the docsearch utility by submitting his own POST requests. More detail on document search service is provided in section 5.3.9.12, The Documentation Search Service.
PUT	The PUT method requests that the enclosed entity be written to the destination. If the PUT refers to an existing resource, the resource may be overwritten. If the request does not refer to an existing resource, the server may create the resource.	The PUT method has been disabled in the evaluated configuration and is not available at the user interface. Methods may be disabled with the appropriate stanza in the */etc/http.conf* file. The RS/6000 Distributed System comes with the appropriate stanza in the */etc/http.conf* file.
DELETE	The DELETE method requests that the origin server delete a resource.	The DELETE method has been disabled in the evaluated configuration and is not available at the user interface. Methods are disabled by appropriate stanza in the */etc/http.conf* file. The RS/6000 Distributed System comes with the appropriate stanza in the */etc/http.conf* file.
TRACE	The TRACE method is used to invoke a remote, application-layer loop- back of the request message.	This interface is not security relevant as it only allows a user to see what is being received at the other end of the request chain and use that data for testing or diagnostic information.

A client request message to a server specifies the method to be applied to a resource. Methods indicate the action to be performed on the resource identified. The following methods are available in HTTP v1.1.

5.3.9.10.3 Responses

After receiving and interpreting a request message, a server responds with an HTTP response message. Server responses are in 1 of 4 categories:

- Informational - Request received, continuing process
- Success - The action was successfully received, understood, and accepted
- Client Error - The request contains bad syntax or cannot be fulfilled
- Server Error - The server failed to fulfill an apparently valid request

5.3.9.10.4 Entity

Request and Response messages may transfer an entity if not otherwise restricted by the request method or response status code. An entity consists of entity-header fields and an entity-body.

5.3.9.11 Web Server

Lotus Domino Go Webserver is a scalable, high-performance web server. It can be extended to include security, site indexing capabilities, and advanced server statistics reporting tools. The file sets that are part of additional Webserver packages are listed in Table 5.34. In the evaluated configuration only the basic Webserver executables have been installed.

Table 5.34. Lotus Domino Go Webserver Packages. *Only the Webserver executables and corresponding English message catalog have been installed in the evaluated configuration.*

Package	File Sets	Description	Installed?
Internet server.base	internet_server.base.admin	administration files	No
	internet_server.base.doc	documentation gifs and HTMLs	No
	internet_server.base.httpd	executables	Yes
	internet_server.msg.en_US.httpd	corresponding English message catalog,	Yes
Internet server.java	internet_server.java.jdk	JDK 1.1.2 for AIX	No
	internet_server.java.servlets	files for java servlet support	No
	internet_server.java.XX	files for java servlet support for XX	No
internet_server.security.common	internet_server.security.common	security files with 56 bit encryption for use outside of North America.	No
internet_server.security.us common	internet_server.security.common	security files with 128 bit encryption for use within North America.	No
NetQ	NetQ.cgi.base	Search Engine CGI executables and icons	No
	NetQ.cgi.XX	Search Engine CGI executables and icons for XX	No

The browser (Netscape) provides users with an interface to request services from the web server and to display the servers responses. The web server runs as the root identity to bind to privileged TCP port 80. The web server forks a copy of itself to run as **nobody** on behalf of the invoking user.

5.3.9.12 The Documentation Search Service

The Documentation Search Service provides a Webserver search form that allows users to search HTML online documents. The service includes a search engine and a CGI layer. The CGI layer is stored in and run by a forked copy of the Webserver on a documentation server computer running as **nobody.**

When the CGI layer is called by an application, it automatically generates a search form inside the browser, passes the user's search query to the search engine, and displays the search results on results pages. A default generic form is provided. However, using commands a user can customize the search form to change things in the form such as the title, text, graphics, and book volumes (indexes) that are searched.

A search is performed for a user calling the DS_FORM CGI program that is installed in the Webserver cgi-bin directory on the documentation server. If you call this program without specifying any parameters, it will, by default, return the generic global search form. A user may, however, call this program with individual parameters or a parameter file as discussed in the next two sections.

5.3.9.12.1 Parameters Method for Calling a Custom Search Form

Table 5.35. Custom Search Passed Parameters. *Users may customize their search through parameter to the doc search cgi script.*

Parameter	Description	Default
columns	Tells the search form CGI how many indexes to display on each line of the search form page.	3
config	Instead of including parameters in your HTML link that customize your search form, you can just specify the name of a configuration file (*config=filename.lang*) that contains your customization information.	None
indexes	Instructs the search form CGI that indexes (search files) to display for selection. When you give a list of indexes to display the search form will contain only those indexes that are in the list and can be found on that documentation server. Any index specified that is not found on the documentation server will not be displayed. This will create a link called "Search" in your documents. When this link is clicked, it will open a generic search form with only the indexes specified available for searching if they exist.	All
lang	Instructs the search form CGI in which language to display the search form.	English
results_title	Instructs the search form CGI what to display as the title of the search results page.	"Search Results"
title	Instructs the search form CGI what to display as the title of the search form page.	"Documentation Search Service"

To include parameters in the HTML link, after the URL of the search form CGI add a question mark (?) followed by an ampersand (&) separated list of parameters and their corresponding values. Table 5.35 describes the parameters users can include in the link command to customize your form.

5.3.9.12.2 Configuration File Method for Calling a Custom Search Form

Instead of listing a long list of customization parameters in the HTML link that calls your search form, you can create a configuration file and insert all of your customization information inside that file. The user creates the configuration file. Read permission must be set for **nobody**. The configuration file path is sent to the web server. The web server reads the file and uses its contents as if they were input as parameters in the command line. There is no local storage of the user's configuration file.

Table 5.36. Custom Search File Parameters. *Users may customize their search by passing a configuration file to the document search cgi script.*

Parameter	Description	Default
columns	Tells the search form CGI how many indexes to display on each line of the search form page.	3
indexes	Instructs the search form CGI that indexes (search files) to display for selection. When you give a list of indexes to display the search form will contain only those indexes that are in the list and can be found on that documentation server. Any index specified that is not found on the documentation server will not be displayed. This will create a link called "Search" in your documents. When this link is clicked, it will open a generic search form with only the indexes specified available for searching if they exist.	All
lang	Instructs the search form CGI in which language to display the search form.	English
results_title	Instructs the search form CGI what to display as the title of the search results page.	"Search Results"
search_page	Redirects the web browser to another URL without changing the HTML link in the document. Outside of the evaluated configuration this may be used to allow users to search the documentation at some remote site instead. In the evaluated configuration, however, the search_page directory specified in the HTTP configuration file overrides this command. All references are relative to the specified directory. Untrusted users can not modify the specification of this directory. Trusted users are instructed not to modify the specification of it.	None
title	Instructs the search form CGI what to display as the title of the search form page.	"Documentation Search Service"
search_top	Replaces the default HTML header of the search form page with the HTML code between the search_top_begin and search_top_end tags. If this is specified the title parameter (or default) is ignored.	None
search_bottom	Replaces the default HTML footer of the search form page with the HTML code between the search_bottom_begin and search_bottom_end tags.	None

Parameter	Description	Default
results_top	Replaces the default HTML header of the results page with the HTML code between the results_top_begin and results_top_end tags.	None
results_bottom	Replaces the default HTML footer of the results page with the HTML code between the results_bottom_begin and results_bottom_end tags.	None

5.3.9.12.3 Documentation Search Service

The parameters of *columns* and *lang* are not security relevant as they only affect the format of the search or results pages that are presented to the user. The only language available in the evaluated configuration is English.

The parameters of *title, search_top, search_bottom, results_top,* and *results_bottom* are not security relevant as they only allow the user to specify the text that appears in the header or footer of his search or results screen. Any text will be accepted and output in the specified area.

The parameters of *config, indexes* and *search_page* are security relevant.

In specifying the name of a configuration file a user may specify any file at all (e.g., */localhost/etc/security/passwd*). This will cause the web server (running as **nobody**) to attempt to read the referenced 'configuration file.' If the file is not readable by **nobody** then access permission is denied and an error is returned to the user. Security is preserved since the web server process will only gain access to the file if the file is publicly readable.

The audit trail provides adequately detailed records to trace the failed read operation back to the individual (LUID) for whom the web server was attempting the access. For remote requests, the LUID of the invoking user is passed to the server as a part of the TCP handshake for accepting the connection. This is a non-standard implementation of UNIX. The protection mechanisms on TCP/IP are explained in section 5.3.8.10, TCP/IP and UDP/IP Protection Mechanisms. The audit event of the failed read operation is dependent upon the contents of earlier audit records as well as attributes which were inherited from the parent process. Interpreting the actions of a process requires that the administrator understand the actions performed by each system call that generated an audit event and the information that is not explicitly contained within the audit record itself. These other audit records describe the process's parent/child relationship, current working and root directory, and the relationship of network client and server processes. The TFM explains how to interpret each of these audit records in the context of the collection of events and provides a worked example of tracking a user action through a network connection. The collection of audit events recorded in association with the failed read operation together with the explanation, instructions, and worked example in the TFM is sufficient for associating the user identity with the failed read operation.

As stated in the table above, the parameter *indexes* instructs the search form CGI that indexes (search files) to display for selection. When you give a list of indexes to display the search form will contain only those indexes that are in the list and can be found on that documentation server. Any index specified that is not found on the documentation server will not be displayed. In the

evaluated configuration any request for an index is limited to indexes within a directory specified in the *http.conf* file. This directory contains only publicly readable indexes.

The search_page parameter allows a user to specify another directory to search for index files. In the evaluated configuration this parameter is superceded with the directory specified in the *http.conf* file.

5.3.9.13 X Windows

X Windows is a graphical user interface that allows a user to make requests to an X server by executing **X** client software. The X server interacts with the local keyboard, mouse, and display to service those requests. The X server is the software that manages one or more displays, one keyboard, and one mouse. The **X** client is a program that generates protocol requests using Xlib and displays on the screen, taking input from that keyboard and mouse. A client sends drawing and information requests to the server, and the server sends back to the client user input, replies to information requests, and error reports.

The X Window system is not limited to a single client interacting with a single server. The X server owns the underlying hardware resources and performs actions mentioned above on behalf of clients by way of protocol requests. The X server can only be started from the native console (not possible from a telnet session). The user starting the X server must be logged in to own the resource for that session, and must own the lft (low function terminal) for the device. There may be several clients interacting with a single server, which is the case when several applications are displaying on a single screen. A single client can also communicate with several servers.

There are three different mechanisms used to facilitate communication between the X server and **X** client software in the standard AIX product: UNIX domain sockets, AIX shared memory transport, and Internet domain sockets. In the evaluated configuration, only one local inter-process communication channel - UNIX domain sockets - can be used for the client/server connection. Unix domain sockets require that the client and server to be on the same host. While it is possible in standard AIX for the client to run on a different host connected over the network via Internet domain sockets, it is not permitted in the evaluated configuration. For non-root users remote access attempts are not recognized as only the root identity can connect to the necessary ports (port 6000-6255). It is possible for an administrator to use Internet domain sockets, but the TFM instructs administrators to not enable use of Internet domain sockets in the evaluated configuration environment.

The user must login at the console of one of the machines in the distributed system and execute XINIT to start the X Windows session. A user can specify permissions to allow other users access to that user's X Windows session over UNIX domain sockets. The SFUG contains instructions and warnings to users as to the ramifications of granting such access permissions to other users and recommends that the settings be 600. The X server and client both have the identity of the user that initiated the X Windows session.

The evaluated configuration contains command line parameters to allow a user to grant or deny access to members of the user's current effective group or to all users. The permissions parameter is interpreted as three octal values specifying read, write, and execute permissions, similar to the standard UNIX permission bits. The execute bit is ignored. Communications between the client and server are always bi-directional. The only meaningful values for any of the three sets of owner, group, and other are 0 and 6. A value of 6 grants read and write access and a value of 0 denies both read and write access.

Access to the X server via the UNIX domain socket is restricted according to permissions. The X server does not create nor bind to the standard UNIX domain socket if permissions are not specified. This results in this mechanism being unavailable to any user. Otherwise, when the user specifies a permissions set other than 0, the server removes the UNIX domain socket so that a new socket special file may be created with the correct permissions.

For the RS6000 Distributed System, the X server is in the TCB because it is used by the administrator and is required to work correctly. The X server only executes with the identity of the user that initiated it and is not a setuid root program. X Windows does not constitute a TCB interface as there is no instance where the **X** client is in user space and the X server is in the TCB. The X server has no privileges, even when invoked by a system administrator.

During the login process, the Terminal State Manager (**TSM**) program sets the permissions on the physical devices so that the current user has exclusive rights to the devices. Object reuse is addressed because these devices are single character, non-buffered devices, and the X server initializes all its internal storage when it starts up. Once a log out is performed the **TSM** sets the permissions to the devices to 600, making them available for the next user to login.

5.3.9.14 timed

The timed daemon is used to synchronize the clocks of the machines that make up the distributed system. The master timed server is contained on the system that stores the shared administrative databases for the distributed system. When timed is started on a client, it queries the master server to receive the correct time. timed is started from the *rc.tcpip* initialization script on each host. Client and server run as setuid root and utilize privileged UDP port 525. Only the root identity can synchronize the clocks, but users can query the daemon for the time (read-only operation). timed reads the master time and adjusts the time on the host in four minute intervals. The interval time is adjustable using the timed configuration file.

5.3.10 Identification and Authentication

User identification and authentication in the RS/6000 Distributed System includes all forms of interactive login (e.g., using the telnet or FTP protocols) as well as identity changes through the su command. These all rely on explicit authentication information provided interactively by a user.

5.3.10.1 User Identification and Authentication Data Management

To ensure that the RS/6000 Distributed System is administered as a single entity, the system maintains a single administrative database on a Network File System (NFS) server, referred to as the administrative master server. The remaining hosts import the administrative data from the master server through ordinary NFS client operations. The administrative files that must be shared are specified in Table 5-1 in section 5.2.8, TCB Databases.

The system maintains a consistent administrative database by making all administrative changes only on the designated NFS server and exporting the database files to all the other computers in the system. A user ID on any computer refers to the same individual on all other computers. In addition, the password configuration, name-to-UID mappings, and other data are identical on all hosts in the distributed system.

Administrators, through the WSM administrative interface, perform changes to the files that constitute the administrative database.

Users are allowed to change their passwords by using the **PASSWD** command, which is a setuid program with the owning userid of 0. This configuration allows a process running the **PASSWD** program to read the contents of */etc/security/user* and to modify the */etc/security/passwd* file for the user's password entry, both which would ordinarily be inaccessible to a non-privileged user process. Users are also forced to change their passwords at login time, if the password has expired.

The mounting of the master happens during the RS/6000 Distributed System boot sequence, and the system is not in the secure state until a successful mount of the administrative data occurs. If the mount fails, the system is inaccessible as it hangs on the mount and will remain so until the administrator corrects the problem on the host causing the mount to fail. Once the problem is resolved and the mount can occur successfully, the system will complete booting.

5.3.10.2 Common Authentication Mechanism

AIX includes a common authentication mechanism which is a subroutine used for all activities that create a *user session*, including all the interactive login activities, batch jobs, and authentication for the SU command.

The common mechanism includes the following checks and operations:

- Check password authentication
- Check password expiration
- Check whether access should be denied due to too many consecutive authentication failures
- Get user security characteristics (e.g., user and groups)

The common I&A mechanism identifies the user based on the supplied user name, gets that user's security attributes, and performs authentication against the user's password. The authenticate subroutine receives the clear text password entered by the user as a parameter and compares it against the encrypted form stored in */etc/security/passwd*. A result of success indicated by a 1, or

a failure indicated by a 0, is returned to the Terminal State Manager (**TSM**) program which continues the login process. See section 5.3.11.1, The **LOGIN** program, for further details.

5.3.11 Interactive Login and Related Mechanisms

There are nine mechanisms for interactive login and similar activities:

- the standard **LOGIN** program for interactive login sessions on the console of a user's local host;
- the telnet protocol and the rlogin protocol for ordinary interactive login sessions on any host in the system;
- the rsh, RCP and the rexec protocol for remote shell, copy, and single command executions;
- the WSM administrative interface;
- the FTP protocol for interactive file transfer;
- and the **SU** command for changing user identity during a session

All of these mechanisms use the common authentication mechanism described above, but only those that create normal interactive sessions use the standard **LOGIN** program; others implement special-purpose types of sessions.

5.3.11.1 The Login Program

The **LOGIN** program establishes interactive user sessions. In AIX, login is part of the Terminal State Manager (**TSM**) program. This program prompts for a user identity and authentication (i.e., password), and validates them using the common authentication mechanism described above. Authentication prompting may also be suppressed when appropriate (e.g., rsh).

If the validation fails, the prompts are repeated until the limits on successive authentication failures are exceeded. Each failure is considered an event that may be audited.

Login establishes a user session as follows:

1. Assigns a session identifier
2. Sets exclusive access for the controlling terminal to the process logging in
3. Calls the common authentication mechanism to check validity of the password provided for the account being accessed, and gains the session security attributes
4. Sets up the user environment
5. Checks for password expiration and if so, prompts for password change
6. The process's user and group identities are changed to those of the user
7. User is changed to his or her home directory
8. Invokes the user's default shell

The **LOGIN** program is always invoked with open file descriptors for the controlling terminal, used when prompting for identity and authentication information, and passes control to the user's shell

when the session is created. At this point, the user session is established, the user environment is set up, and the program replaces itself, using the ***exec*** system call, with the user's shell).

5.3.11.2 Network Login

After an initial login on the console of any host in the distributed system, access to other hosts within the system may occur through one of seven network protocols: telnet, rlogin, rsh, rcp, rexec, WSM, and FTP.

5.3.11.2.1 Login with telnet

The telnet protocol always requests user identity and authentication by invoking the LOGIN program, which uses the common authentication mechanism. A user can change identity across a telnet connection if the password for another account is known.

5.3.11.2.2 Login with rlogin

The rlogin protocol includes user identity as part of the protocol information passed from host to host. User is not permitted to switch identity between hosts using ***-l*** option.

5.3.11.2.3 Command execution using rsh

The rsh protocol includes user identity as part of the protocol information passed from host to host. User is not permitted to switch identity between hosts using ***-l*** option.

5.3.11.2.4 Command execution using rcp

The RCP protocol includes user identity as part of the protocol information passed from host to host. User is not permitted to switch identity between hosts using ***-l*** option.

5.3.11.2.5 Command execution using rexec

The rexec protocol always requires the user to enter a valid user identity and password. The authentication is performed by invoking the common authentication mechanism directly rather than by invoking login. User can change identity if password is known.

5.3.11.2.6 WSM login

WSM is the primary administrative interface for the distributed system. When WSM is run in remote mode, a dialog box pops up and prompts for the user name and password. The WSM server performs explicit authentication based on that user identity and password supplied by the WSM client.

The authentication is performed by invoking the common authentication mechanism directly rather than by invoking login.

5.3.11.2.7 File transfer using FTP

The FTP protocol is used to create a special type of interactive session that only permits file transfer activities. An FTP session is validated and created directly by the FTP server, which then executes all the user requests directly, as opposed to invoking a user-specified program.

The FTP server invokes the **LOGIN** program that uses the common authentication mechanism to validate the user identity and password supplied through FTP protocol transactions. User can change identity if password is known.

5.3.11.3 User Identity Changing

Users can change identity (i.e., switch to another identity) using the SU command. When switching identities, the login UID is not changed, so all actions are ultimately traceable in the audit trail to the originating user. The primary use of the SU command within the RS/6000 Distributed System is to allow appropriately authorized individuals the ability to assume the root identity. In this system the capability to login as the root identity has been eliminated. In the */etc/security/user* file, login to root is set to false for all users and SU is set to true for administrators. This allows an administrative user to login under his/her real identity, then SU to the root identity.

The SU command invokes the common authentication mechanism to validate the supplied authentication.

5.3.11.4 Login Processing

Permissions on the device special files control access to exclusively used public devices. When a user successfully logs in at the local attached terminal, the **TSM** program changes the ownership of */dev/lft0, /dev/kbd0, /dev/mouse0* and */dev/rcm0* to the login UID of the user and sets the permissions on these devices to octal 600. */dev/lft0* is a logical device that provides the users interface to the keyboard, mouse, and graphics adapter. At system initialization, */dev/lft0* grabs the keyboard, mouse and graphics adapter devices.

The */dev/kbd0* device contains two channels for communication between the keyboard and the device driver. Only one channel is active at any given time. The */dev/lft0* device registers for the first channel when the system boots. The second channel is reserved for the X server. The permissions on the */dev/kbd0* device restrict that only the user who is logged in on the console can access this device. The logged in user could open the second channel, because he/she has permissions. This would redirect the users own keyboard device. This would pose no threat to the operation of the system. The worst thing that would happen is that the login process would not be able to regain access to the */dev/kbd0* device and no other users would be able to login on the console device until the host was rebooted.

The */dev/mouse0* device contains only one channel, which is grabbed by the */dev/lft0* device on system startup. Attempts to open additional instances of the */dev/mouse0* device will result in an error message.

The login process executes a ***revoke*** to invalidate any open file descriptors for */dev/lft0* held by a previous user. The ***revoke*** call modifies the file descriptors entry in the system open file table, causing further attempts to access the device special file based on that file descriptor to return "bad file descriptor". This ensures that the new login session is isolated from any previous login sessions.

5.3.11.5 Logoff Processing

When a user logs off, all files that were opened by the login shell are closed. Files and devices that were opened by background tasks remain open. However, a background job that had access to the console loses that access prior to the next user's login as stated above.

The ownership of */dev/lft0*, */dev/kbd0*, */dev/mouse0,* and */dev/rcm0* is returned to root when the logoff occurs, with octal permissions set to 600.

5.3.12 Batch Processing

Deferred processing provides a method for commands to be executed while the user is not physically logged in to the system. Batch processing is the facility for jobs to be scheduled and executed. A job is a command or set of commands that performs an action on the system.

There are two methods to perform batch processing, through the use of the AT command and the CRONTAB command and file. The AT command allows a user to schedule jobs from the command line, and the CRONTAB facility uses a file to specify when jobs will execute.

If a job is recorded for processing, and the user who created the job is deleted from the system before the job has run, the job will not execute. If a user account is disabled, any cron and at jobs will execute. The TFM contains instructions for canceling deferred processing requests when a user is removed from the system or disabled.

When a user submits a batch job, the user's effective UID, effective GID, supplementary groups, login ID, and audit information are recorded. These values are used when the batch job is run, ensuring that audit information is associated with the user whose session originated the request, while using the permissions associated with the effective UID.

5.3.12.1 Batch Processing User Commands

The AT command schedules the execution of jobs based on a time value. A user submits a job; the job is placed in a queue, and is executed when the system time reaches the start time for the job. An AT job can be submitted with the *now* parameter, which causes the job to run immediately. All output from the job is mailed to the user unless output is purposely redirected.

The BATCH command executes a job when the system load level permits. BATCH is a script that calls the AT command with parameters specifying that the command should be placed in the batch queue versus the AT queue.

The CRONTAB command runs setuid to root and allows a user to submit, edit, list or remove CRON jobs. The individual crontab files are stored in the */usr/spool/cron/crontabs* directory. Users have no access to this directory. Each *crontab* file is owned by root and group cron, with permissions: rw- --- ---. The CRONTAB command is setuid to root to allow the untrusted user to perform submit, edit, list or remove operations on their own *crontab* file.

Table 5-37. The Crontab File. *The crontab file contains a list of jobs to be executed in batch mode.*

minute	hour	day_of_month	month	weekday	command

When inserting a job into the *crontab* file, the user may use the first five fields to specify when the job will run, as well as the job's frequency. If the user specifies an asterisk for any of the first five values, then all allowed values will be substituted. All allowed values for minute would be every minute in an hour, hour includes 00 through 23, day of month includes day 1 through 31, month includes 1 through 12, and weekday includes Sunday through Saturday, entered as 0 through 6. Each user of the system can create and modify their own *crontab* files, as long as they are not explicitly denied access to the CRON facilities.

5.3.12.2 Batch Processing Daemon

The cron daemon facilitates the execution of batch jobs. CRON is started from the rc initialization file of each host in the system. Each host maintains its own cron daemon, crontab files and batch processing queues. When a job is executed by CRON, a shell is started in the users home directory that executes the command. CRON maintains a log file that reports its activities, as well as two separate directories that store queued jobs, one for at jobs and one for crontab files. A named pipe is used to send messages to the cron daemon when new jobs are submitted with the CRONTAB or **AT** commands.

The cron daemon references four different files for **AT** and CRON access control decisions (*at.allow, at.deny*, *cron.allow* and *cron.deny*).

Table 5-38. Control Files Referenced by the cron Daemon. *The* CRONTAB *daemon accesses four different files to determine whether a job can be submitted via* **AT** *or* **CRON**.

File	Purpose
at.allow	Specifies which users are allowed to submit jobs using at.
at.deny	Specifies which users are not allowed to submit jobs using at.
cron.allow	Specifies which users are allowed to use the **CRONTAB** command.
cron.deny	Specifies which users are not allowed to use the **CRONTAB** command.

If *at.allow* does not exist, and *at.deny* exists, any user not listed in *at.deny* is allowed to use the AT command. If neither *at.allow* nor *at.deny* exist, only the root identity may use the AT facilities. The same statement is true for CRON.

5.3.13 Printer Services

Printer services are the commands and daemons that handle the queuing and servicing of print jobs, both local and remote. Printer services include a collection of user commands for adding and deleting print jobs and a collection of administrative commands and daemons to process queued print jobs.

A print queue is an ordered list of print requests for a specific printer. The */etc/qconfig* defines the print queues available for use and the **BACKEND** program used to process the print jobs. The **PIOBE BACKEND** command is invoked for the processing of local jobs, and the **REMBAK** command is invoked for the processing of remote jobs.

5.3.13.1 Daemons Used with Printing

The **qdaemon's** job is to track the print requests, and ***fork*** and ***exec*** the **PIOBE** or **REMBAK** commands to process a print job.

The lpd daemon is a remote print server that monitors TCP port 515 for incoming print requests from remote clients. lpd has a directory *(/var/spool/lpd*) where incoming requests are placed while awaiting further processing. lpd will only accept print requests from remote machines that are listed in */etc/hosts/equiv* or */etc/hosts.lpd*.

The lpd daemon calls **ENQ** to queue a print job that is received over the network, on the local system. From that point on, the print job follows the path of a local print job.

5.3.13.2 Interface for Queuing Print Jobs

There are three different methods for queuing print jobs: **QPRT, LP,** and **LPR**. All three of these commands create and queue a print job by calling the **ENQ** command. **ENQ** can also be invoked directly to submit print jobs.

QPRT is the standard method provided with AIX, **LP** and **LPR** are provided for those familiar with System V and BSD methods of printing.

5.3.13.3 Interface for Manipulating Print Queues

The following tasks may be performed on individual print jobs: holding and releasing the job, moving a job between queues, canceling a job, changing the priority, or checking the status.

On the local machine, a user may affect only a print job that belongs to him/her. When a print job is queued, a file is created in the queue directory. The user owns the file. When a command is executed that attempts to manipulate a particular job, a check is performed to verify that the UID that owns the file (queued print job) is the same as the UID that is attempting to perform an action on the job or root.

The owner of the print job on the remote host is the user lpd and the group printq. The job description file for the print job contains the UID of the user that submitted the job. QDAEMON uses this information when a request is received to cancel a job.

5.3.13.4 Print Job Processing

Print jobs can be printed to a local printer or to another host in the distributed system.

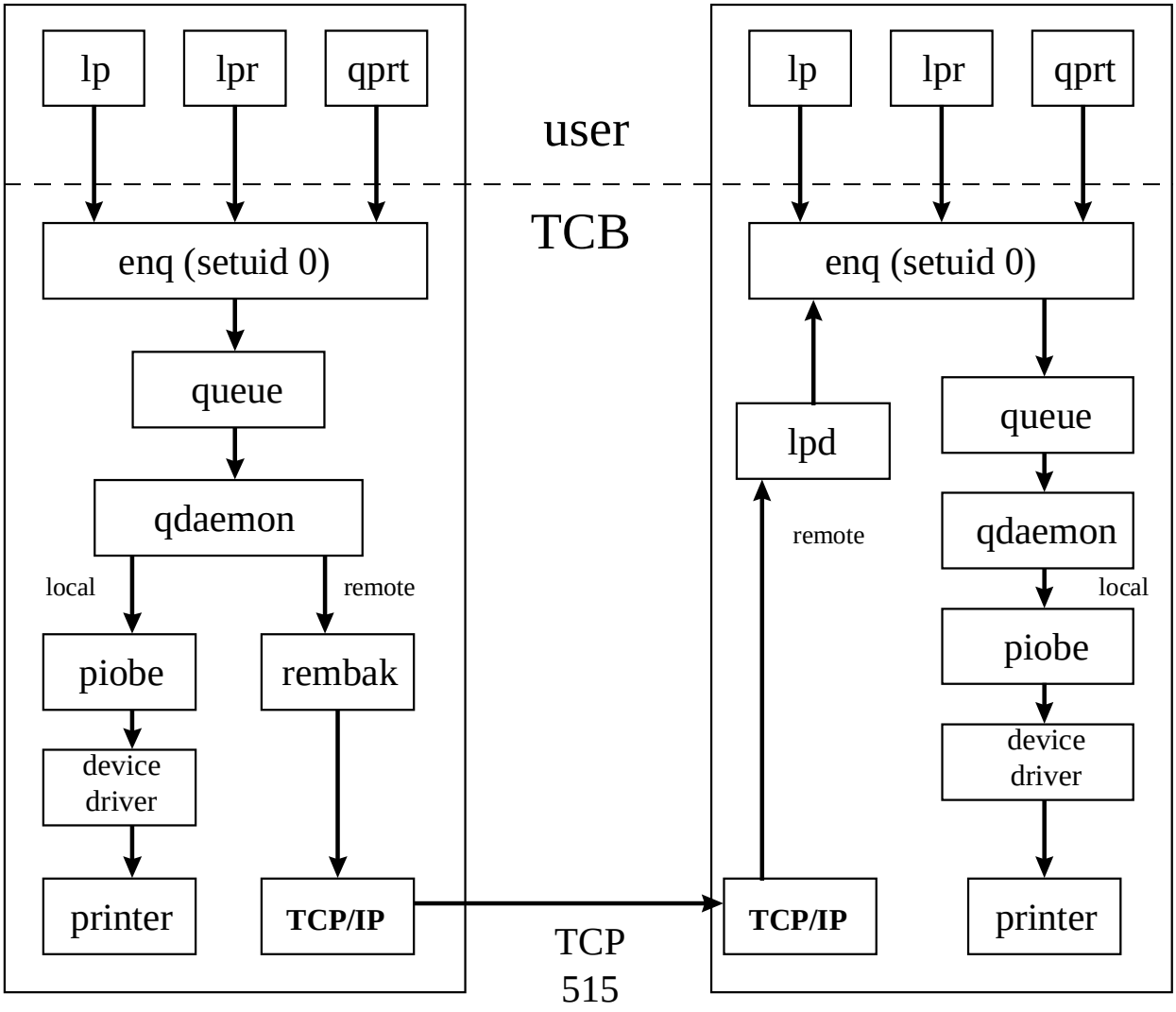

Figure 5.5: Local and Remote Print Job Processing

5.3.13.4.1 Local Print Job Processing

- A front-end print command such as QPRT, LPR, or ENQ initiates the request to the appropriate queue on the local system.
- The qdaemon passes the request to the PIOBE backend on the print server.
- The piobe backend formats the data stream for printing on the specified printer and sends the job to the printer via the device driver.

5.3.13.4.2 Distributed Print Job Processing

- A front-end print command such as QPRT, LPR, or ENQ initiates the request to the appropriate queue on the local system.
- The **qdaemon** on the local system processes the request by passing it to the **rembak** backend program.
- The **rembak** program transmits the print job to a remote server via TCP/IP.
- On the remote server the lpd daemon monitors port 515 for remote print requests.
- When the lpd receives a remote print request, it places the job in the appropriate local queue.
- The print request is then processed by the **qdaemon** on the print server.
- The **qdaemon** passes the request to the **piobe** backend on the print server.
- The **piobe** backend formats the data stream for printing on the specified printer and sends the job to the printer via the device driver.

5.3.14 Mail

Electronic mail provides a method for users of the distributed system to exchange electronic messages. The mail system consists of two components: a user interface and a message transfer agent.

Each host has a directory (*/usr/spool/mail*) containing incoming mail. The permissions on */usr/spool/mail* are drwxrwxr-x, owned by user bin, group mail. Each user of the system may have a file in that directory for their system mailbox. A user's system mailbox is owned by the user and owned by the group mail. The user's mailbox provides the following permissions: rw-rw- ---. A user can see that other user's mailbox files exist, but they have no other access to those files.

If a user has not received mail on a particular host, they will not have a system mailbox on that host. The system mailbox is created when the user receives their first message on the host, and is a normal file.

The .forward file is a text file located in the user's home directory, that contains a list of e-mail addresses that incoming messages will be forwarded to. If the .forward file exists for a user, incoming mail is automatically delivered to the address specified in the file.

The .forward file can also specify a command to forward messages. Commands executed as a result of the .forward file are executed using the UID of the user. The VACATION command is commonly used to reply to all incoming mail with a message stating that the current user is on vacation and will return to the office soon.

5.3.14.1 Mail Reading

The MAIL program allows users to create, read, forward and archive messages. The MAIL program reads the **.mailrc** configuration file on execution. This file allows the user to customize the MAIL program with a number of options.

The use of the mail user-interface programs (**MAIL, MAIL, MAILX**) require no special privilege, because the user is only accessing his assigned, unique local system mailbox. If a user sends a message from one of the user-interface programs, **SENDMAIL** is invoked to perform the message delivery.

5.3.14.2 Mail Delivery

Users of the distributed system can send and receive messages with local users as well as with users on other hosts. The **SENDMAIL** program is used to deliver messages locally and between nodes in the system. **SENDMAIL** is invoked for two different purposes. The first is to act as a daemon, and the second is to facilitate delivery of a mail message, either locally or remotely. The version of **SENDMAIL** used with the distributed system is 8.8.8.

SENDMAIL references a configuration file, */etc/sendmail.cf*, to identify the variables that are used in the processing of mail and how addresses are processed for delivery. Mail headers are created for incoming messages by **SENDMAIL**, using definitions contained within */etc/sendmail.cf*. The definitions consist of a series of text and macros that are translated when a message is received on a host.

The RS/6000 Distributed System TFM provides warnings to prevent the administrator from modifying the *sendmail.cf* file from the evaluated version to prevent assigning of "trusted users" for **SENDMAIL**. Trusted users, as defined within **SENDMAIL**, have the capability of sending mail as anyone in the system.

When **SENDMAIL** is executed as a daemon, it is started by the *rc.tcpip* script during system initialization, and is bound to TCP port 25 and awaits TCP communications using the SMTP protocol.

When **SENDMAIL** is executed as a user program, set-user-ID to root is invoked to allow **SENDMAIL** to access another user's mail file for local delivery, or open a TCP/IP connection to a remote host on port 25 for remote delivery using SMTP. When **SENDMAIL** is invoked as a user program, it accepts RFC 822 formatted e-mail messages. The interface to **SENDMAIL** collects text through standard in, and formats this text as an e-mail message.

For local delivery, **SENDMAIL** formats the message and passes it to **BELLMAIL**. **BELLMAIL** performs the operation of appending the message to the user's system mailbox.

SENDMAIL contains a number of command-line options for modifying the way mail is delivered. **SENDMAIL** acts as the interface for mail delivery in the distributed system, while the mail program acts as the user's interface to his system mailbox.

Users can access **SENDMAIL** directly, but cannot affect the operation of the **SENDMAIL** daemon or perform any other tasks that would not be performed automatically for them during the normal delivery of a mail message. The **SENDMAIL** daemon is a trusted process, executing as UID 0, which prevents untrusted users from accessing it by any means other than the defined interface for sending mail.

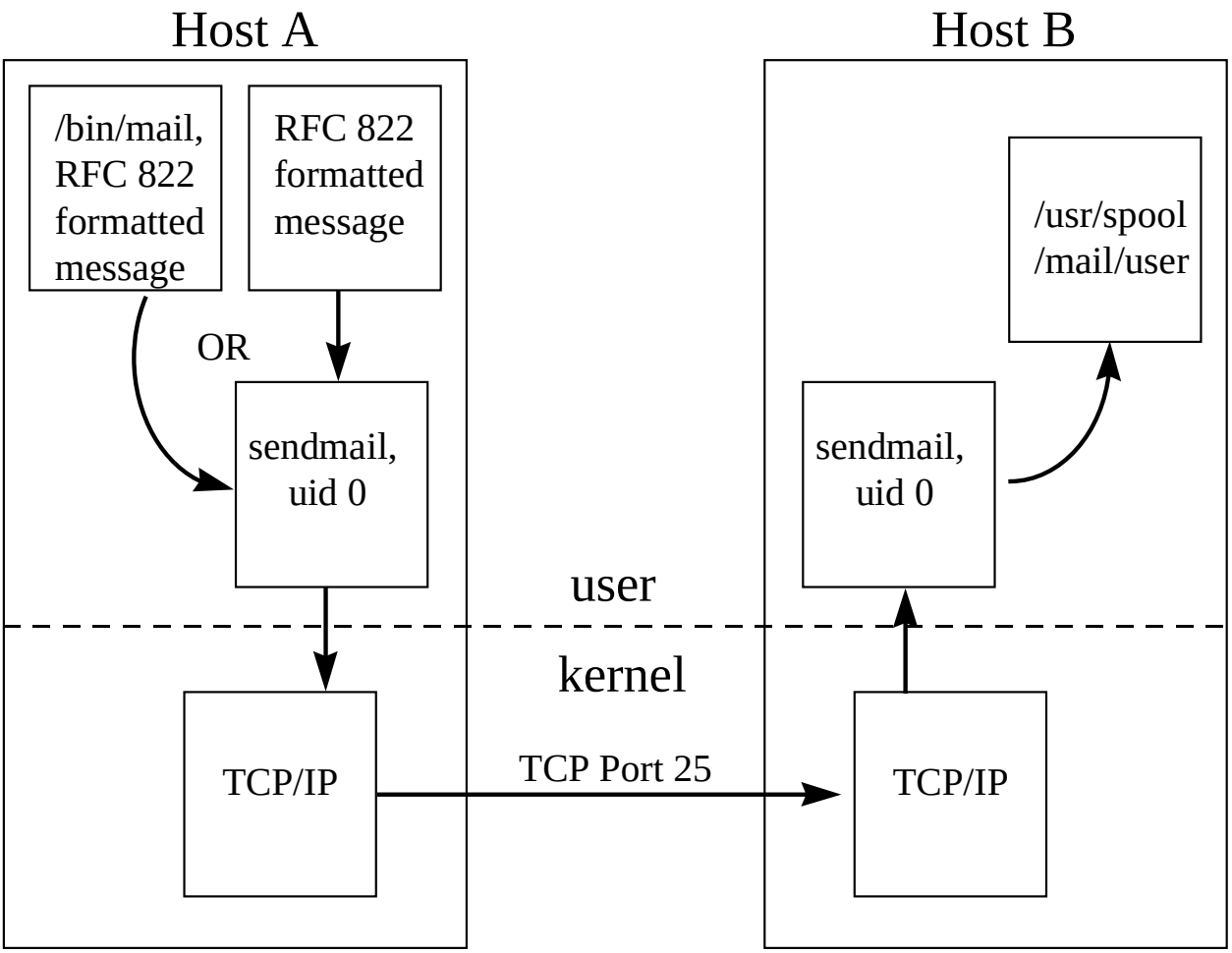

Figure 5.6: Mail Communication between Host A and Host B

An untrusted user can invoke **SENDMAIL** with the -C command line option. The -C option provides a method for specifying an alternate configuration file to **SENDMAIL**. The -C option does not effect the correct operation of the **SENDMAIL** daemon that is bound to TCP port 25 and receives mail from other hosts. The **SENDMAIL** daemon is a separate instance of **SENDMAIL** from the one executed by the user. The -C option can only be used to effect the copy of **SENDMAIL** that a user executes to deliver a mail message.

When the -C option is used, a setuid is performed to the real UID of the user. This blocks **SENDMAIL** from delivering local mail or remote mail. For local mail, **SENDMAIL** is executing as the user, and cannot shell and run **BELLMAIL** to append to a local mailbox. For remote mail, **SENDMAIL** cannot connect to the remote host on TCP port 25 because the access control allows only connections from a UID of 0. Any parameters that are specified differently in the alternate **SENDMAIL** configuration file are not security relevant, because the user is not able to cause the delivery of mail using the alternate config file.

The security of **SENDMAIL** is improved by implementing access control on TCP port 25 and not allowing external network connections to the distributed system. The connection between **SENDMAIL** on the local host and **SENDMAIL** on a remote is a TCB to TCB operation, because

access control on TCP port 25 prevents connections from user ID's other than zero. Access control on port 25 prevents fraudulent mail messages that could be created by an untrusted user opening a telnet session to a remote host and spoofing an SMTP session. The distributed system may not be attached to any external networks or external network devices, such as hubs or routers, and may not be attached to any machines that are not running the evaluated configuration. Blocking physical access to and from other types of machines closes the network and eliminates the potential for intrusions from outside sources.

Table 5-39. sendmail command line options. ***SENDMAIL*** *has a number of command line options. The following options are listed because they are interesting in regards to whether an untrusted user or root can utilize them.*

Command Line Option	Description	User Allowed	Root Allowed
-bd	Invokes **SENDMAIL** running as a daemon, in background mode. Normal users cannot invoke **SENDMAIL** as a daemon running in the background, because the **SENDMAIL** command blocks this option.	N	Y
-bD	Invokes **SENDMAIL** running as a daemon, in foreground mode.	N	Y
-bp	Prints a listing of entries in the mail queue.	Y	Y
-bh	Prints the persistent host database.	N	Y
-bH	Purges the persistent host database.	N	Y
-bi	Builds the alias database from information defined in the /etc/aliases file.	N	Y
-bs	Uses SMTP to collect mail from standard input.	Y	Y
-bt	Starts the **SENDMAIL** command in address test mode.	Y	Y
-bv	Starts **SENDMAIL** with a request to verify user IDs provided in the address field of the command.	Y	Y
-C file	Starts **SENDMAIL** using an alternative configuration file.	Y	Y
-f	Sets the name of the sender of the mail.	N	Y

5.3.15 Auditing

This section discusses the implementation of auditing in the evaluated configuration. The data structures and formats are discussed first, followed by how audit is controlled, a description of bin mode auditing, the programs used to post process the audit data, the programs used to review audit data, audit file protection, and finally the potential for audit data loss.

Audit data is generated separately on each host in the distributed system and may be managed and analyzed either separately on each host in the distributed system, or merged and analyzed on a single system. AIX includes tools for pre-selecting and post-selecting audit data, viewing audit trails, and merging multiple audit trails into one file.

5.3.15.1 Audit Record Format

The audit record consists of a header that contains information identifying the user and process who generated the record, the status of the event (success or failure), and the CPU id for the system. The CPU id field allows the administrator to differentiate between individual machines when merging the contents of multiple audit trails. An optional variable length tail contains extra information about the event, as defined in */etc/security/audit/events*.

Table 5-40. Audit Event Record Format. *The audit record is a fixed length record that contains information about the user who caused the event and whether the event was created due to a success or failure. The audit record is defined in /usr/include/sys/audit.h.*

Magic number for audit record.	
The length of the tail portion of the audit record.	
The name of the event and a null terminator.	
An indication of whether the event describes a successful operation. The values for this field are:	
0	Indicates successful completion.
1	Indicates a failure.
>**1**	An ***errno*** value describing the failure.
The real user ID; that is, the ID number of the user who created the process that wrote this record.	
The login ID of the user who created the process that wrote this record.	
The program name of the process, along with a null terminator.	
The process ID of the process that wrote this record.	
The process ID of the parent of this process.	
The thread ID.	
The time in seconds at which this audit record was written.	
The nanoseconds offset from time. (used during bin recovery to ensure proper record ordering)	
CPU identifier.	

5.3.15.2 Audit Control

Audit control consists of the files used to maintain the configuration of the audit subsystem and a description of the AUDIT command and its associated parameters.

Table 5-41. Audit Control Files. *The audit control files maintain the configuration for the auditing subsystem.*

Audit Control File	Description
/etc/security/audit/config	Defines whether bin mode auditing is enabled and available classes
/etc/security/audit/events	Defines audit events available for use on the system
/etc/security/audit/objects	Contains a list of the objects whose access will be audited
/etc/security/audit/hosts	Contains a mapping of CPU ids to hostnames in the distributed system
/etc/security/audit/bincmds	Contains the post-processing command or commands for bin mode auditing
/etc/security/user	Specifies which classes will apply to the current user account

There are two different types of audit event selection: per-user and per-object. Per-user auditing allows the administrator to specify specific classes of audit events that will be recorded for that user. Each process stores a copy of the audit classes that apply to that user as part of the process table. An audit class is a subset of the total number of audit events available.

Per-object auditing allows the administrator to specify file system objects that will be audited. These objects can be audited based on accesses of a specified mode (read/write/execute) and record the result of the access attempt (success/failure).

The **AUDIT** command is used to start and stop the auditing subsystem, to temporarily switch the auditing subsystem on or off, and to query the audit subsystem for the current audit parameters. The **AUDIT** command is started from the host's rc initialization script, as stated in the RS/6000 Distributed System TFM.

The on and off parameters of the **AUDIT** command enable and disable audit, without modifying the current configuration that is stored in the kernel. The on parameter can have an additional parameter, panic, which causes the system to shut down if bin data collection is enabled and records cannot be written to one of the bin files. The bin mode panic option can also be specified in */etc/security/audit/config*.

When the **AUDIT** command is issued with the shutdown parameter, the collection of audit records is halted, and all audit configuration information is flushed from the kernel's tables. All audit records are flushed from the kernel's buffers and processed. The collection of audit data is halted until the next audit start command is entered.

When the **AUDIT** command is issued with the start parameter, the following events occur:

- the */etc/security/audit/config* file is read
- the */etc/security/audit/objects* files is read and the objects that will be audited based on access are written into kernel tables
- the audit class definitions are written into kernel tables from */etc/security/audit/config*
- the **auditbin** daemon is started, depending on the options in */etc/security/audit/config*
- auditing is enabled for users specified in the user's stanza of the */etc/security/audit/config* file
- auditing is turned on, with panic mode enabled or turned off, depending on what mode is specified in */etc/security/audit/config*

5.3.15.3 Audit Record Generation

Audit record generation begins with the detection of an event, and follows the record as it advances to storage.

Event detection is distributed throughout the TCB, both in kernel and user mode. Programs and kernel modules that detect events that may be audited are responsible for reporting these events to the system audit logger. The system audit logger is part of the kernel, and can be accessed via a system call for trusted program auditing, or via a kernel procedure call for supervisor state auditing.

The audit logger is responsible for constructing the complete audit record, including the identity and state information and the event specific information. The audit logger appends the record to the active bin. A bin is a file that is used to store raw audit records before they are processed and stored in the audit trail.

5.3.15.4 Audit Record Processing

Audit record processing includes a description of bin mode auditing and the backend processors that are utilized by the audit subsystem.

5.3.15.4.1 Bin Mode Auditing

When Bin mode auditing starts, two separate bin files are allocated to store raw audit records by the **auditbin** daemon. When one bin file fills, the daemon switches to the other bin file and invokes the processing command specified in */etc/security/audit/bincmds* to empty the full cache file. When that operation is complete, **auditbin** notifies the kernel that it is permitted to reuse the cache file. This mechanism of switching and emptying audit bins continues so long as auditing is enabled. The size a bin file may reach before being considered full is defined in */etc/security/audit/config*.

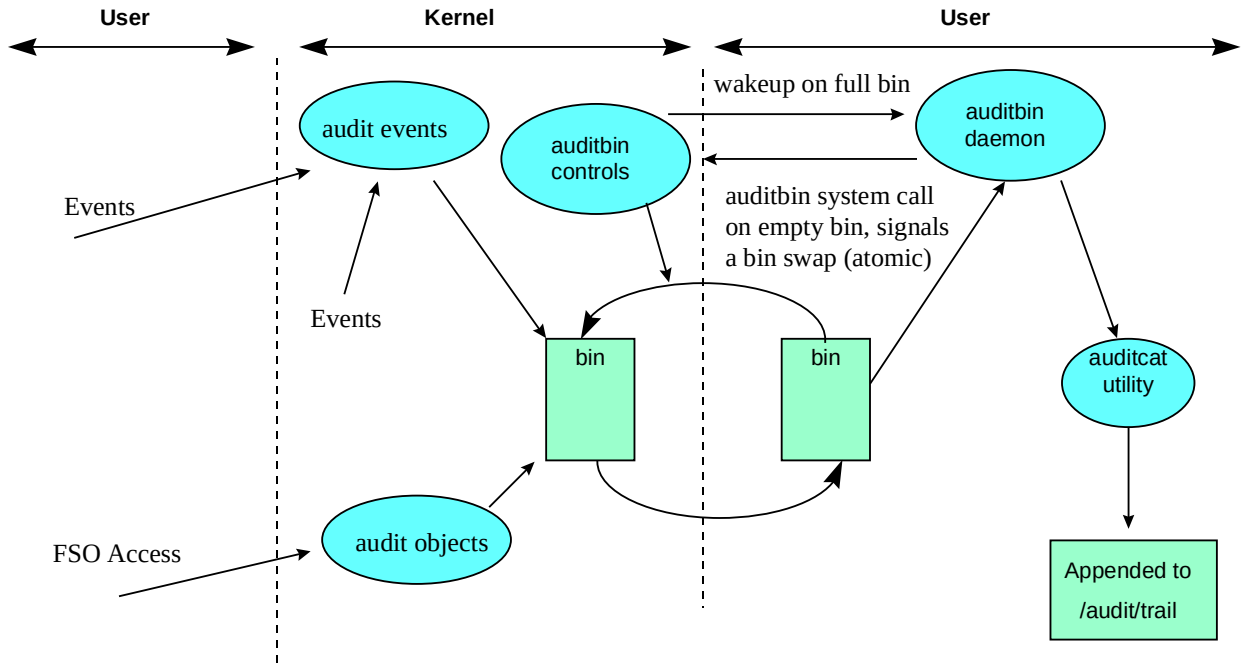

Figure 5.7: Bin mode auditing

The following format describes the header and tail of the bin file. A bin file begins with a header followed directly by a tail. As audit records are stored in the bin, the tail is continuously overwritten and rewritten at the end of the file. When the bin file reaches the full size, it is cleared out, and the bin header and tail records are reset at the beginning of the file. At this point the file is ready to be reused when the other bin fills.

Table 5-42. Bin Header and Tail Record Format. *The bin header and tail record are used to mark the beginning and end of an audit bin, and to hold the variable length tail for an audit event record. These structures are defined in /usr/include/sys/audit.h.*

The magic number for the bin.	
The version number for the bin (2)	
Indicates whether the bin describes the audit trail head or tail:	
0	Identifies the bin header.
1	Identifies the bin end (tail).
2	Identifies the trail end.
The (unpacked) length of the bin's records. A nonzero value indicates that the bin has a tail record.	
The current length of the bin's record (might be packed).	
The time at which the head or tail was written.	
CPU id	

5.3.15.4.2 Backend Audit Processors

There are two backend processors available for use: AUDITCAT and AUDITSELECT. The backend processor writes the raw audit records to the system audit trail or to a specified file after manipulating them.

Bin mode auditing makes use of AUDITCAT and AUDITSELECT. The result of AUDITCAT or AUDITSELECT can be directed to a file for permanent audit storage.

5.3.15.4.2.1 auditcat

The AUDITCAT command reads audit records from standard input or from a file, and processes the records and sends them to standard output or to the system audit trail.

5.3.15.4.2.2 auditselect

The AUDITSELECT command can be used as both a pre-processing and post-processing tool. As a pre-processing tool, the AUDITSELECT command serves the same purpose as AUDITCAT, but adds the ability to specify conditions that an audit record must meet. This allows a system to be configured to save audit records that relate to login in one file, and audit records that relate to file access in a separate file.

AUDITSELECT utilizes an expression to apply against the current audit record. The expression consists of one or more terms joined by the logical operators && (and), || (or) and ! (not). Each term in the expression describes a field, a relational operator and a value.

The following is an example expression to select all the FILE_Open events:

event==FILE_Open

The event field identifies that AUDITSELECT should query based on the name of the event. The operator is equal and the name of the event is FILE_Open.

Table 5-43. Available Fields. *The available fields are used to build expressions with* **AUDITSELECT.**

Field	Definition
event	Name of the audit event
command result	Status of the audit event. The value of the result field must be one of the following: OK, FAIL, FAIL_PRIV, FAIL_AUTH, FAIL_ACCESS, or FAIL_DAC. FAIL matches all other error codes.
login	ID of the login user of the process that generated the audit event.
real	ID of the real user of the process that generated the audit event.
pid	ID of the process that generated the audit event.
ppid	ID of the parent of the process that generated the audit event.
tid	ID of the kernel thread that generated the event.
time	Time of day the audit event was generated.
date	Date the audit event was generated.
host	Hostname of the machine that generated the record. The reserved name UNKNOWN can be used to match any machines that are not listed in the */etc/security/audit/hosts* file.

5.3.15.5 Audit Review

Two different commands exist for the review of audit records in the distributed system: **AUDITPR** and **AUDITSELECT**.

The **AUDITPR** command formats audit records to a display device or to a printer for review. The **AUDITPR** command also allows the administrator to select which of the fields to include in the output as well as the order to display them. The fields available for inclusion with the output of the **AUDITPR** command are listed in the following table.

Table 5-44. Available Fields From AUDITPR. *These fields are are available for output from* **AUDITPR.**

audit event	user's login name	event status	time the record was written	command name	real user name	process ID	ID of the parent process	kernel thread ID	name of the host that generated the audit record	event specific tail data

The default values are the audit event, the user's login name, the audit status, the kernel thread ID and the command name

AUDITSELECT allows the administrator to build an expression that will be applied to the stored audit records. The details of the **AUDITSELECT** command are listed in section 5.3.15.4, Audit Record Processing.

The **AUDITMERGE** command provides a method of combining multiple audit trail files into a single audit trail file. These multiple files can come from different hosts in the system, providing a centralized audit analysis function. As the two files are processed, the record with the oldest time stamp that still remains is written into the audit trail. This process continues until there are no more audit records to process. The TFM directs the system administrator to transfer the audit files to be merged to the same host.

5.3.15.6 Audit File Protection

The audit trail files, configuration files, bin files, and the */audit* directory are protected on each system using normal file system permissions. Each audit file grants read access to the root user and the audit group, and write access to only the root user. The RS/6000 Distributed System TFM instructs the administrator that if the cached and permanent audit trails are kept other than in the */audit* directory, then the alternate directory must be protected from access by non-root users.

5.3.15.7 Audit Record Loss Potential

Bin mode auditing is susceptible to the exhaustion of disk space available to the */audit* directory or to a system crash. In the case of a system crash, all data in physical memory is lost, including any audit records that had not yet been flushed to disk. The audit subsystem enforces a 32K byte limit on the size of an individual audit record, and only one audit record can be in transit between a thread and the kernel at any given time.

The RS/6000 Distributed System TFM includes instructions to the administrator to back up all files, including audit data, on a regular basis to avoid the loss of data due to hard disk failures.

5.3.15.7.1 Audit Record Loss Potential for Bin Mode Auditing

The RS/6000 Distributed System provides a panic mode for use with bin mode auditing. The panic mode option halts the host when the current audit bin stops accepting additional records, preventing the unnecessary loss of audit records. This only occurs with the exhaustion of disk space. If a host halts because it cannot collect audit records, the other hosts in the distributed system are not affected, unless the host is acting as the administrative master. The RS/6000 Distributed System TFM contains instructions for enabling panic mode, as panic mode is not enabled by default.

The result of halting the system because panic mode was invoked would be the loss of any audit data presently in the host's memory that had not been written to disk. In addition, audit records could be lost for operations that were underway but had not yet completed generating audit records. This minimizes the damage caused by the lack of disk space, because only the audit records that are currently in memory are lost.

A recovery process for audit bins exists in the evaluated configuration. If either of the bin files is not empty when audit is started, the **auditbin** daemon executes the bin mode post-processing command to process the bins.

The amount of audit data that can be lost in bin mode is minimized by the use of the binsize and bytethreshold parameters in the */etc/security/audit/config* file. The binsize parameter sets the maximum size a bin may reach before the **auditbin** daemon switches to the other bin, and executes the bin mode post-processing command. The bytethreshold parameter sets the amount of data in bytes that is written to a bin before a synchronous update is performed. The RS/6000 Distributed System TFM states that the binsize and bytethreshold parameters should be set to 64K bytes each

to minimize audit data loss. The amount of audit data that could be lost due to a failure in bin mode is the combination of these two files, or 128K bytes.

5.3.15.8 Administrative Auditing

The root user can have any of the classes defined in */etc/security/audit/events* enabled. This provides auditing of all the normally defined system functions. Events exist that audit changes to the TCB databases.

The TCB_Exec event audits actions performed by the root user. The TCB_Exec event is triggered when a process created by root performs an ***exec***. The ***exec*** system call detects if the effective user-id is equal to zero, and writes this audit event if it is enabled. This capability allows for the auditing of all administrative actions on the system.

5.3.16 Initialization and Shutdown

The RS/6000 Distributed System consists of one or more computers, each of which can be in a multi-user mode, single-user mode, or can be powered off.

The order in which computers are brought online is important. The administrative master of the RS/6000 Distributed System must be booted before any other computer in the system can be powered up and used. This is because the administrative databases are NFS-mounted from the master server. Other computers may be booted to perform maintenance, but users will not be able to login. The boot process will not complete for administrative clients until the master server and its databases are available.

The secure initial state for AIX has been reached before display of the login prompt but after a series of initialization related tasks. These tasks include the initialization of privileged ports above 1024 and TCP socket ACLs, the configuration of the NFS subsystem, the mounting of the distributed I & A files, the initialization of audit logging, and the re-execution of the DACINET configuration commands to update protection methods for dynamic services (**mountd, statd** and **lockd**).

5.3.16.1 Boot Methods

In the evaluated configuration, (as described in the TFM) AIX must be booted from a boot image on the local hard disk.

A boot password can be set, but this password is not relied upon for the enforcement of any policies with respect to the evaluated configuration. When the power button is pushed, users are only allowed to boot the system as initially installed. If a user has knowledge of the boot password, the user is still restricted to booting the system using the default media. To boot the system differently requires knowledge of the Privileged Access Password (PAP). The TFM contains installation instructions requiring the administrator to set the PAP and to protect that password from non-root users as this password is critical to maintaining the integrity of the TCB.

5.3.16.2 Boot Image

The file system that is used during the initial boot phase is located in a special logical volume known as the Boot Logical Volume (BLV). The default logical volume name is */dev/hd5*, although this name is not required to be used by the system.

This logical volume has a file system type of boot and is updated by the **bosboot** command. Files that are contained in the BLV are copied from the run-time file system, converted into a compressed file system image and copied to the BLV. Only the administrator may modify the BLV as the file system permissions for the device deny read or write access to any other users. Changes to the RAM file system during the initial boot phase are not copied back to the BLV making the BLV file system image read-only. The RAM file system acts as a mini-root file system during the boot process.

The boot logical volume consists of the following components:

- Boot record
- SOFTROS software used to complement the CHRP open firmware Read Only Storage (ROS) on the CPU board.
- Boot image, consisting of the AIX kernel and a RAM file system.
- Optional base customized area.

It is this boot image (the data on the boot logical volume) that is then used to bootstrap the system. The boot code found in the BLV is used to load the kernel and to mount the RAM file system into memory. This RAM file system is memory-resident and contains all programs that allow the boot process to continue. This file system does not exist outside of the boot process.

Administrators can reconstruct the BLV by bringing the system up in maintenance mode and using the **bosboot** command.

5.3.16.3 Boot Process

During the boot process, the system tests the hardware, loads and begins execution of AIX, and configures devices. As described above, the boot code found in the BLV is used to load the kernel and to mount the RAM file system into memory. Once all of the devices associated with the root volume group are configured, the volume group can be initialized, the real root file system can be mounted and the RAM file system is discarded.

For an individual computer, the boot process consists of the following steps when the CPU is powered on or reset:

1. The CPU initializes and performs its internal Built-In Self-Test (BIST).
2. On the 43P and F50, the CPU then executes the power-on self-test (POST) code.
   On the S70, the Service Processor sets up the hardware as a RS/6000 and runs POST.
3. The CPU executes the CHRP Open firmware (also known as ROS) for the platform.

4. The Open Firmware finds the SOFTROS code in the boot image (*/dev/hd5* by default), and runs SOFTROS to perform additional initialization to provide an environment for AIX. (This is CHRP-compliant code loaded from the system's hard disk storage.)
5. SOFTROS reads in the boot control block from disk storage.
6. The Boot Loader loads the AIX kernel and RAM file system, and transfers control to the kernel initialization code. The kernel creates process 0, which boots the system, then becomes the swapper.
7. The kernel initializes its data structures, and starts the Simple Shell (*/usr/lib/boot/ssh*) as init.
8. Phase 1 Boot. Phases 1 and 2 run from the RAM file system. In Phase 1, the ***ssh*** (init) expands the RAM file system, restores saved device configuration information, and calls the configuration manager ***cfgmgr*** to perform its first phase duties. At various locations throughout the boot process it displays different values to the LEDs to indicate boot status.
9. Phase 2 Boot. The root volume group is initialized, base file systems are mounted, and ***fsck*** is run. If a dump is present from a previous boot, maintenance mode is entered to allow the dump to be captured if required. The base paging logical volume is initialized and paging is started.
10. Phase 3 Boot. Between phase 2 and phase 3, the RAM disk is destroyed and ***init*** is the real init. It runs through */etc/inittab*, the first item being */sbin/rc.boot 3***.** This runs phase 3 of ***rc.boot*** which does initialization using the real file system. The */tmp* file system is mounted and ***fsck*** run. The last phase of ***cfgmgr*** is run, dump devices are defined, lock files are removed and daemons like ***syncd*** and ***errdemon*** are started.
11. At this point control is passed to */etc/rc* to continue booting.
    a. Run ***rc.powerfail***
    b. ***rc*** starts multi-user mode, initialization of volume groups, page space activation, dump file designation, file system check (***fsck***)
    c. Execute ***fbcheck*** to determine if this is the first time AIX has been booted. If it is the first time, a license agreement is displayed.
    d. Start System Resource Controller (***srcmstr***) daemon
    e. Initialize privileged port extension and TCP socket ACLs
    f. init runs ***rc.tcpip*** to start the network daemons
    g. ***rc.nfs*** is executed to configure the NFS subsystem
    h. Executes the script */etc/rc.C2* to mount the distributed administrative files (if this step fails, the system will hang waiting on the NFS mounts to become available; administrative intervention must be performed in order to correct the problem and make the machine available for multi-user mode). Once this step has been performed, the system has all of the defined users and known hosts available via the NFS mounts.
    i. Start ***cron***
    j. Start ***getty***
    k. Start ***startsrc***

l. Last step is re-executing the ***DACinet*** configuration commands since name resolution is performed with the NFS mount and port protection definitions may contain hostnames and user IDs. This also configures port protection for the dynamic RPC servers.

12. ***getty*** puts up a login prompt.

5.3.16.4 Shutdown

The SHUTDOWN or REBOOT commands can be used by the administrator to bring services and the system off-line gracefully (these commands are restricted to the root identity). The TFM warns administrators to only use the SHUTDOWN command and not the REBOOT command in the evaluated configuration.

During the default shutdown, users are notified of an impending system shutdown with a message. After the administrator-specified number of second's elapse, the system stops the accounting and error logging processes and writes an entry to the error log. The SHUTDOWN command then runs the KILLALL command to end any remaining processes and runs the SYNC command to flush all memory resident disk blocks. Finally, it UNMOUNTS the file systems and calls the HALT command. Shutdown is not complete until users receive a shutdown completion message. Attempting to restart or turn off the system before the shutdown completion message is displayed may result in file system damage.

The SHUTDOWN command supports a number of flags to control how the system is to be brought down. By default it will warn users, wait one minute, terminate active processes, synchronize the file systems, and halt the CPU as described above. An administrator has the option to restart the system immediately after shutdown, perform a fast shutdown (bypassing display of messages to users and bringing the system down as quickly as possible), and bring the system down to maintenance (single-user) mode.

5.4 TCB Support

This section identifies TCB software components in the RS/6000 Distributed System that are in the TCB because they have the ability to affect correct operation of the TCB, but which have no specific security responsibilities. These components do not implement or support security policies or maintain security-relevant databases.

5.4.1 Internal Daemons

The TCB contains numerous processes that are always running but that do not have specific security responsibilities. Some run entirely in the kernel, called **kprocs**; others are started during initialization and run as ordinary processes. None are security relevant.

kprocs are primarily concerned with buffering and synchronization. Additionally, each CPU has an idle process that runs when no other process needs the CPU. The non-kernel daemons support buffering, synchronization, and logging.

5.4.2 Uninteresting Trusted Processes

Some TCB programs in the RS/6000 Distributed System are in the TCB only because they are required for performing administration or system initialization. These have no responsibility for implementing specific security policies or maintaining specific security databases, but they are in the TCB because they are used in a privileged environment and have the potential for affecting the correct operation of the TCB.

Because they have no security responsibilities other than not acting maliciously, they are termed uninteresting. The RS/6000 design documentation identifies all these programs. Typically they are file-oriented utility programs such as **LS, VI, GREP**, and **CAT**.

Some uninteresting TCB programs have privileges that are active when an administrative user uses the program. Although this privileged identity allows the command to override normal security policies, they still have no explicit security responsibility even when operating with these privileges. When used by an ordinary user, these programs never have privilege. For a complete list of all trusted processes in the evaluation configuration, see Appendix H, Trusted Programs.

5.4.3 TCB Libraries

Most libraries are included in the TCB simply because they are invoked and hence execute as part of one or more TCB processes. The same library, executing as part of a non-TCB user process, is untrusted code. A library can be used both by TCB and non-TCB code, and is trusted when it executes in the context of a TCB process and is untrusted when it runs in the context of a non-TCB process.

Any TCB program that runs without privilege must ensure that it is only running with libraries that are explicitly included in the TCB. This is done by providing a TFM warning for administrators not to change the LIBPATH environment variable.

6. TCB RESOURCES

This chapter identifies the RS/6000 Distributed System resources and outlines the security policies applicable to them. This discussion includes all resources that are accessible to subjects operating on behalf of ordinary users and sharable among subjects with different security characteristics. It does not include internal resources (such as internal TCB structures) that are only accessible by trusted subjects; nor does it include resources (such as administrative data) that only administrators may manipulate.

Most resources are subject to security policies that may result in different access being granted to different users or subjects, and are therefore subject to the discretionary access control policy. A few resources (e.g. unnamed pipes) are dynamically shared only among subjects with equivalent security attributes, but because different subjects serially reuse them, it is important to consider them as resources for the purposes of object reuse.

6.1 Resource Characteristics

Table 6-1 characterizes each resource in terms of how an instance of the resource is created, the nature of the Discretionary Access Control policy that applies to the resource, and whether the resource is a named object. The column headings, Create, Named and O.R. Method, are defined in sections 6.1.1, Creation; 6.1.4, Named Objects; and 6.1.3, Object Reuse respectively. Details of the protection policy for resource types are specified in Chapter 7, TCB Policies.

6.1.1 Creation

Each resource is created (or allocated from the system's set of instances) by either an ordinary or administrative user.

User: A subject operating on behalf of an ordinary user may create resource instances. Programming interfaces usable by untrusted subjects may create resources of this type. Although for many, commands are the conventional interface mechanism.

Administrator: Only an administrative user or a TCB subject may create instances of the resource. Although these resources may also be created by a variety of programs, administrators are required to use only the programs (e.g. commands) permitted by procedures defined in the TFM, and are not permitted to exercise arbitrary programming interfaces.

6.1.2 Discretionary Access Control

The RS/6000 Distributed System enforces a discretionary access control policy for all named objects. While the enforced DAC policy varies among different object classes, in all cases the policy is based on user identity. To allow for enforcement of the DAC policy, all users must be identified and their identities authenticated.

Under the generally enforced policy, subjects (i.e., processes) are allowed only the accesses specified by the class-specific policies. Further, the ability to propagate access permissions is limited to those subjects who have that permission as determined by the class specific policies. Finally, a subject with an effective UID of 0 (i.e., the root identity***)*** is exempt from DAC restrictions and can perform any action.

6.1.3 Object Reuse

All resources are protected from Object Reuse (*scavenging*) by one of three techniques: explicit initialization, explicit clearing, or storage management. This section describes how AIX prevents improper reuse of residual data for each class of object identified in section 6.1.4, Named Objects and section, 6.1.5 Storage Objects. It addresses only those objects directly visible at the TCB interface and is not concerned with internal operating system structures. Three general techniques are used to meet this requirement:

- ***Explicit Initialization:*** The resource's contents are explicitly and completely initialized to a known state before the resource is made accessible to a subject after creation. An 'I' in the object reuse column in Table 6-1 indicates explicit initialization.
- ***Explicit Clearing:*** The resource's contents are explicitly cleared to a known state when the resource is returned to the TCB for re-use. A 'C' in the object reuse column in Table 6-1 indicates explicit clearing.
- ***Storage Management:*** The storage making up the resource is managed to ensure that uninitialized storage is never accessible. An 'M' in the object reuse column in Table 6-1 indicates storage management.

6.1.4 Named Objects

In the RS/6000 Distributed System a named object is one whose intended use exhibits the following characteristics:

- may be used to transfer information between subjects of differing user identities within the TCB.
- a specific instance of the object must be requestable by subjects.
- The name used to refer to a specific instance of the object must exist in a context that allows subjects with different user identities to request the same instance of the object.
- The intended use of the object is for sharing of information across user identities.

The named objects in the RS/6000 Distributed System are designated within Table 6-1 with a 'Y' in the "Named" column.

6.1.5 Storage Objects

For the purposes of satisfying the Object Reuse requirement, all identified resources in the RS/6000 Distributed System are considered *storage objects*. Adequate controls are in place such that no resources permit an unprivileged subject to see residual information.

All the resources identified within Table 6-1 are considered storage objects for the purposes of object reuse.

Table 6-1. Object Summary. *Each resource in the RS/6000 Distributed System was analyzed in terms of how an instance of the resource is created, the nature of the Discretionary Access Control policy that applies to the resource, and whether the resource is a* named object.

Resource[9] Type	Create	Named	O. R. Method
Ordinary Files	user	Yes	I,M
Directory	user	Yes	I
Directory Entries	user	No	C
Symbolic Links	user	No	I
Character Device Special Files	admin	Yes	I
Block Device Special Files	admin	Yes	I
Named Pipes	user	Yes	I
Unnamed Pipes	user	No	I
UNIX Domain Socket Special File	user	Yes	I
System V Shared Memory	user	Yes	I
System V Message Queues	user	Yes	I
System V Semaphores	user	Yes	I
Printer Queue Entries	user	No	I
At Queue Entries	user	No	I
Crontab Queue Entries	user	No	I
Process (includes context and address space)	user	No	I,M
Datagrams	user	No	I
Mail Files	user	No	I
UDP datagrams, TCP connections	user	No	I,M
X Windows Resources	user	No	C
Printer DRAM	user	No	C
Frame Buffer	user	No	C

6.1.6 Public Objects

Public objects can be read by all subjects and written only by trusted processes. At the C2 level of trust these objects are not security relevant because they are not named objects and therefore do not require DAC. Furthermore, these objects have no object reuse concerns because they are

---

[9] TCP/IP Ports: These are addresses used to identify TCP services. They do not have the capability of storing information and are therefore not considered objects.

Internet Domain Sockets: This term refers to a socket that is bound to an Internet address and protocol. Sockets contain no information. They are abstractions used in communication, but are not intended for (and cannot be) used for controlling sharing of information. Rather, any sharing that occurs using these mechanisms is of necessity the result of a prior agreement between the two communicating peers. The need for prior agreement is inherent in the definition of the network communication protocols supported by the RS/6000 Distributed System. A socket is a privately owned endpoint for communications; that is they are only sharable by inheritance, and cannot be named or manipulated by processes that do not have access to them already. Because the socket interface and the underlying protocols are defined to provide communication with arbitrary peers, without controls by user identity, DAC is not appropriate for these IPC mechanisms.

public objects. An example of this class of objects for the RS/6000 Distributed System is the system timer. The system timer is set by the TCB and is readable by everyone.

6.2 Users

A ***user*** is a person who has been identified to the RS/6000 Distributed System by an administrator and is authorized to use the systems facilities.

6.2.1 User Roles

There are two types of users within an IBM RS/6000 Distributed System: ordinary and administrative users. Only administrators can create and destroy user accounts. Administrators use the Web-based System Manager (WSM) utility to perform account creation and maintenance operations, as well as other administrative functions required to keep the system operational.

6.2.2 User Attributes

Users have security attributes that govern the set of system facilities they may employ. This section describes those attributes and their maintenance. Other sections, such as those in Chapter 5 that describe the I&A process, describe how the attributes are used in making security decisions.

User attributes include security relevant information (e.g., one way encrypted password, identity, permissions), administrative limitations (e.g. CPU time and memory quotas), and general information (e.g., the user's full name). Some security-relevant user attributes (e.g., one-way encrypted password) are sensitive and may only be viewed by an administrator. All other user attributes are publicly visible.

6.2.2.1 Identity

A user is identified by a human-readable user name, which is administratively assigned. Each user name corresponds with a unique numeric user identity or UID. A one-to-one correspondence exists between the set of user names and the set of UIDs across the distributed system via the NFS-mounted database mechanism described in Section 5.3.10, Identification and Authentication.

Table 6-2. Reserved UIDs.

User Name	User ID	Description
root	0	This user has the ability to perform all system administrative tasks. Users cannot login to this account,. They must login to their own account and ***su*** to the root identity.
nobody	-2	This identity is used in the evaluated configuration by the HTTP server. While the evaluated configuration includes NFS, "nobody" is not used in that capacity for the evaluated configuration.[10] Since the evaluated configuration is a closed distributed network with one name spaced across the system, identity mapping is not performed and has not been analyzed as part of this evaluation.
daemon	1	This user does not own any files, programs or directories in the evaluated system.
bin	2	This user owns a number of system binaries and the directories which hold them.
sys	3	This user owns the crontab file for user *sys* which performs no operations.
adm	4	This user owns the **/var/adm/streams** directory as well as the **/var/adm/wtmp** system account file. This user also owns the crontab file for user *adm*.
	5	Not used in the evaluated configuration but reserved for standard AIX.
	100	Not used in the evaluated configuration but reserved for standard AIX.
lpd	9	This user is used by the printer queuing subsystem. Users can neither login nor su to this account.
	6	Not used in the evaluated configuration but reserved for standard AIX.

All security-relevant decisions are made on the basis of the user identity. A user must enter the user identity, along with a valid password, to gain access to the system. The user identity is the basis of all DAC decisions, and the user identity is stored within audit records to maintain user accountability.

The user identity root (i.e. UID 0) is the only privileged account within the evaluated configuration. It is possession of the root identity and password that distinguishes an administrative user from an ordinary user. Possession of the root identity allows a user access to administrative data and files that are critical to the correct operation of the TCB. For the evaluated configuration, it is not possible to login directly as the root identity**.** The administrator must first login using a valid ordinary user account and then perform an su to change their identity to the root identity.

Reserved user IDs are used throughout AIX to indicate ownership of various system files, directories and programs. The RS/6000 Distributed System was evaluated with all of these reserved administrative identities disabled for login, and except for the root identity, disabled for su.

6.2.2.2 Groups

In addition to the user identity, a user is associated with one or more groups, each of which also has an administratively assigned group name that translates to a numeric group identity or GID for system internal use. The set of GIDs is the same across the distributed system via the NFS-

[10] Typically the "nobody" identifier is used to indicate a file that was created by root on an **NFS** file system that does not map the client's root user to the server's root user.

mounted database mechanism described in Section 5.3.10, Identification and Authentication. The primary use of groups is in DAC decisions.

Table 6-3. Reserved GIDs.

Name	Group ID	Description
system	0	This group controls a number of administrative functions, such as shutting down the system, mounting file systems. The *system* group additionally owns a number of system files and directories.
nobody	-2	This group is not used in the evaluated configuration. Even though the evaluated configuration includes NFS, it is a closed distributed network and as such group identity mapping is not performed and has not been analyzed as part of this evaluation. There is one group name space across the distributed system and so group name resolution is explicit.
staff	1	This group does not own any files or programs in the evaluated configuration. It exists for compatibility with the non-evaluated version of AIX.
bin	2	This group owns most of the unprivileged system binaries as well as the directories which hold those files.
sys	3	This group owns a number of system configuration tools primarily in the **/usr/lib/ras** and **/usr/sbin** directories.
adm	4	This group owns a number of directories that contain administrative data, such as system logs and accounting information.
	5	This group does not own any files or programs in the evaluated configuration. It exists for compatibility with the non-evaluated version of AIX.
mail	6	This group owns the programs and directories that are associated with local mail processing. The program **BELLMAIL** is used to read and deliver local mail.
security	7	This group owns the programs, directories and files that contain the I&A information and file management. Many of the programs are restricted to group *security* execution only and may only be executed by the *root* user in the evaluated configuration.
cron	8	This group owns the programs, directories and files that contain and administer the atjob and crontab information.
printq	9	This group owns the programs, directories and files that contain and administer print jobs, print queues, batch queues and other queuing related functions. Many of the programs are restricted to group *printq* execution only and may only be executed by the *root* user in the evaluated configuration.
audit	10	This group owns the programs, directories and files that contain and administer the auditing functions of the system. Many of the programs are restricted to group *audit* execution only and may only be executed by the *root* user in the evaluated configuration.
ecs	28	This group does not own any files or programs within the evaluated system. It exists for compatibility with the non-evaluated version of AIX.
	100	This group does not own any files or programs in the evaluated configuration. It exists for compatibility with the non-evaluated version of AIX.
	20	This group does not own any files or programs in the evaluated configuration. It exists for compatibility with the non-EVALUATED version of AIX.

6.2.2.3 Authentication

Associated with each user identity is a hashed (one-way encrypted) password for login authentication. In addition, there are many password quality parameters (governing such things as password length, prohibited passwords, number of alphabetic characters) and password expiration parameters (governing when the password expires, how long before it can be reused, how long after the password has expired it can be changed).

Each user entry includes account usage parameters governing account expiration, times of use, from what terminals, the time and location of the last login and batch job execution, and the number of login failures since the last successful login attempt. There exists a set of parameters that govern whether the *su* utility can be used to change to this user identity, and who may perform such changes. In addition, each user entry has a flag that may be set by an administrator to disable a user account.

6.2.2.4 Audit Control

Each user entry has a list of audit classes specified for the audit events to be recorded for the user. All auditable events (as pertains to the audit class specified for that user) performed by processes created on behalf of that user will generate audit records.

6.2.3 User Database

User security attributes are stored in */etc/security/users*, */etc/security/passwd*, and */etc/passwd*. Only administrators can read */etc/security/user* and */etc/security/passwd*. All users can read */etc/passwd* so that untrusted software can obtain needed information, but only administrators can write to it. As explained in section 5.3.10, Identification and Authentication, the contents of */etc/security* (and other configuration data) are stored on one host of the distributed system, and are NFS-mounted on all other hosts to maintain consistent administrative information.

In a correctly administered system (*i.e.*, one that is administered following the guidance in the TFM), every user will have one entry in */etc/passwd*, one in */etc/security/user*, and one in */etc/security/passwd*. None of these files will have entries for users who do not also appear in the other files.

Table 6-4 describes the security relevant user attributes. These security attributes can only be set or viewed by administrators using the WSM utility. AIX's file locking mechanisms prohibit the modification of these database entries by more than one administrator.

In addition to the above parameters, */etc/security/environ* defines the initial set of environment variables for each user, and */etc/security/limits* defines resource limits for each user (*e.g.*, the maximum file size, the maximum amount of memory available to processes). For non-administrative users, these parameters are not security relevant. For administrators, the environment variables must be set appropriately (*e.g.*, the variables that control search paths for programs and shared libraries must point to only IBM-provided directories and files).

Table 6-4. User Security Attributes.

Attribute	Description	Stored in:
user name	A one to eight character unique identifier.	*/etc/passwd*
UID	A unique numerical value associated with the user name. This value is associated with any session created for the user.	*/etc/passw*
primary GID	The initial GID associated with the user.	*/etc/passwd*
supplemental GIDs	Additional GIDs associated with the user name.	*/etc/group*
current password	The user's current password.	dummy value in */etc/passwd*, real password in encrypted form in */etc/security/passwd*
password quality parameters	A set of parameters that govern the minimum password length, the minimum number of alphabetic characters, a dictionary of prohibited passwords.	*/etc/security/passwd*
authentication method	A set of parameters governing how users are to be authenticated. In the evaluated configuration there is only one method.	*/etc/security/user* and */etc/security/login.cfg*
password expiration parameters	A set of parameters that govern when the password expires, how long before it can be reused, how long after the password has expired it can be changed.	*/etc/security/user*
account usage parameters	A set of parameters that govern when the account expires, what times of day and days of the week it can be used, what terminals it can be used from.	*/etc/security/user* and */etc/security/login.cfg*
user change parameters	A set of parameters that govern whether the ***su*** utility can be used to change to this user identity, and who may perform such changes.	*/etc/security/user*
audit classes	A list of audit classes for which audit events are to be recorded for the user.	*/etc/security/audit/config*
umask	Contains default file permissions.	*/etc/security/user*

6.3 Subjects

In the RS/6000 Distributed System, processes are the only subjects. Every process has a unique process identity (PID) relevant to the host on which it executes. A process consists of an address space, with one or more threads of execution, on a specific host machine. Processes cannot execute across multiple hosts, so it is the combination of the PID and host name that ensure uniqueness of process identity across the distributed system. PIDs may be reused over the lifetime of the system, but there are never two simultaneously existing processes with the same PID on the same host.

6.3.1 Identity Attributes

The following security-relevant user and group identifiers are associated with each process:

- **Login UID**: is the UID of the account used to log in to the system. It never changes as long as the user is logged in. This is the UID stored in audit records.
- **Real UID and GID**: is the UID of the account for the user of the system. It changes whenever a the user successfully issues the su command. It can also be changed by a process running as root issuing the setuid subroutine.
- **Effective UID and GID**: is the same as the real UID/GID unless the process has executed a setuid program.
- **Saved UID and GID**: These values are set to the values of effective UID and GID at the time of process creation. For unprivileged programs, these are the UID and primary GID of the user. For setuid or setgid programs, these values are used to restore the effective UID/GID upon return from a setuid program.
- **Supplementary group Identities**: Additional GIDs associated with the user (beyond the real GID). Set as part of the login process; normally not changed. There can be up to 32 supplementary GIDs. The supplementary GIDs are used in access control decisions.

6.3.2 Additional Process Security Attributes

Additional security relevant attributes of processes include:

- **Process state:** The current state for each process includes the set of associated threads (where the default is one thread per process) and the attributes enumerated below. For each thread, state information includes the address of the next instruction to be executed (*i.e.*, the program counter or Instruction Address Register) as well as all general purpose and floating point registers.
- **Open file descriptors:** Every process has a list of open file descriptors that are accessible by the process.
- **Session ID:** Every process has an associated session ID, which is the process ID of the process group leader. This value is used to determine which process will receive unsolicited notifications of process termination.

6.3.3 Privilege Attributes

In the credentials structure[11] for each process are four privilege sets as well as all relevant identification information (real, effective, and saved user and group ids and login user id) for that process.

[11] The credential structure is discussed in section 5.3.2, Process Management.

The privilege sets are, respectively:

- Effective
- Inherited
- Bequeathed
- Maximum

Processes running as the root identity (directly or via setuid-root) have all privilege bits enabled in all four of the above privilege sets. Processes running as a non-root identity have none of the privilege bits set in any of the privilege sets and have no ability to gain privilege when a system is properly installed and administered in accordance with the TFM. See Section 7.2, Privileges for additional information regarding the privilege policy for AIX.

6.4 File System Resources Security Attributes

This section describes the common attributes for file system object types. Then it describes the attributes that are special to, or specially interpreted for, different object types.

6.4.1 Common File System Resource Attributes

This section describes the attributes that are common to all types of file system objects and that have common meaning for those types. Their general meaning and behavior are described here; details are addressed in the per-type subsections that follow. Note that the security control policies based on these attributes are described in detail in Chapter 7. When policies are mentioned here, it is only done to give some context to the descriptions.

- **Owning user ID:** This is the effective user ID of the subject that created the object. In general, it governs the ability to manipulate object attributes.
- **Owning group ID:** This is either the effective group ID of the creating subject or the owning group ID of the directory in which the object was created.
- **Base Permissions Bits:** This is a 9-bit string specifying three possible modes of access (‘read,’ ‘write,’ and ‘execute’) to the object for each of three classes of users (owner, owning group, and other). Not all access modes are relevant or interpreted identically for all object types.
- **Extended Permissions:** This is a list of user IDs and/or group IDs that specifies access given to an object for those named users and groups. These extended permissions, also called Access Control Lists or ACLs, are optional and may only be set by the object owner. Extended permissions, when present, are used in combination with permission bits to determine access to an object.
- **TCB Bit:** Any file system object may have this attribute set to indicate its membership in the TCB. This bit is not used nor consulted in the evaluated configuration for security policy enforcement.

6.4.2 Ordinary File Security Attributes

Ordinary files include both ordinary text and program files. In addition to the common attributes, file objects have the following attributes:

- **setuid:** causes a process to change effective user ID when the program is executed.
- **setgid:** causes a process to change effective group ID when the program is executed.

6.4.3 Directory and Directory Entry Security Attributes

In addition to the common attributes, directory objects have the following special attributes:

- **setgid:** Causes objects created in the directory to inherit the directory's group ID, rather than the effective group ID of the creating process.
- **savetext:** used to restrict deletion rights. If the *savetext* bit is set on a directory, then files (and subdirectories) can only be deleted by the processes with an effective UID equal to the owning UID of the file (or subdirectory). This is in contrast to directories where the savetext bit has not been set, where any user with the *write* permission to the directory can delete a file or subdirectory. This special case is an adjunct to the access control policy, and is not part of the access control policy itself.

Directory entries have no security attributes on their own. All access control derives from the attributes of the directory of which they are a part. They are treated as distinct resources solely for the purpose of Object Reuse protection.

6.4.4 Symbolic Link Security Attributes

Symbolic links have no additional security attributes.

6.4.5 Device Special File Security Attributes

Device Special Files, character and block, have no additional security attributes.

6.4.6 Named Pipe (FIFO) Security Attributes

FIFOs have no additional security attributes.

6.4.7 Unnamed Pipe Security Attributes

Unnamed pipe objects have no additional security attributes. Further, because they are created in the "open" state, and are not namable in the file system, the permission bits of these objects have no meaning.

Although unnamed pipes are not part of the file hierarchy, in the sense of having a pathname like all other file system objects, they are part of the file system in the sense of being implemented

similar to files because they are manipulated with the same file descriptor interfaces as other file system objects.

6.4.8 Socket Special File (UNIX Domain) Security Attributes

UNIX domain sockets have no additional security attributes.

6.5 Inter-Process Communication Resources Security Attributes

All System V IPC have associated with them the creator's UID and GID as well as the current owner's UID and GID. Base permission bits are maintained and used in access control decisions. Extended permission bits are not available on these types of objects.

6.5.1 System V Shared Memory Security Attributes

A shared memory segment refers to a portion of memory that may be mapped into the address space of one or more processes. When a shared memory segment is created a unique identifier is allocated to it that is used by other non-TCB subjects as a key (external name) to reference it.

6.5.2 System V Message Queues Security Attributes

A Message queue is like a mailbox holding typed messages. Messages may be retrieved by type or FIFO order. It allows processes to communicate with each other by placing and retrieving messages on a specified message queue. The message queue ID gives them a unique external name.

6.5.3 System V Semaphores Security Attributes

A semaphore is an IPC mechanism used to relay some condition to all participating processes. For example semaphores can be used for process synchronization, or for sharing of a resource between different processes. Semaphores are implemented in sets consisting of one or more semaphore values. Each semaphore has a unique identifier that can be used as an explicit external name to refer to it by different processes.

6.6 Queuing System Related Resources Security Attributes

6.6.1 Printer Queue Entry Security Attributes

Printer Queue entries are created by users submitting print jobs with the **LPR** command. The security attribute of the printer queue entries is the user ID of the submitting process.

6.6.2 At-Job Queue Entry Security Attributes

At-job queue entries are created by users submitting batch jobs with the AT command. The security attribute of the at-job queue entries is the Real User ID of the submitting process.

6.6.3 Crontab File Security Attributes

The security attribute for the *crontab* is the owning user ID.

6.7 Miscellaneous Resources Security Attributes

6.7.1 Processes

Processes are the active agents (subjects) of the RS/6000 Distributed System. However, they may also be viewed as objects, operated on by other processes, and as such have attributes that govern the ability of other processes to perform those operations: the real and effective user IDs.

In addition, processes have numerous other attributes that control their capabilities when performing operations. See section 6.3, Subjects for a complete list.

There are two interfaces that operate on processes:

- the PS command and related commands, for displaying process status and attributes;
- the ***kill*** system call, for sending signals;

The relevant attributes of process objects with respect to these operations are the effective user and group IDs, and the real user and group IDs.

6.7.2 Datagrams and TCP Connections

Datagrams and TCP connections have no specific security attributes and are only interesting for the purposes of object reuse.

6.7.3 Mail Files

Each user mailbox is an ordinary file, and has the security attributes associated with ordinary files. When a user mailbox is created, the user owns it.

Users cannot create their own mailbox. Mailboxes are created by the BELLMAIL program at the time mail is sent to a user for which no mailbox already exists. DAC permissions (755) on the */usr/spool/mail* directory enforce this restriction.

6.7.4 Printer DRAM

The printer DRAM has no specific security attributes and is only interesting for the purpose of object reuse.

6.7.5 Frame Buffer

The frame buffer has no specific security attributes and is only interesting for the purpose of object reuse.

7. TCB POLICIES

This chapter describes how the RS/6000 Distributed System Trusted Computing Base (TCB) controls the resources described in Chapter 6. It begins with a description of the policies, and then provides a per policy, per resource description of how the policies are applied.

7.1 Identification and Authentication

The software involved in user identification and authentication is discussed in section 5.3.10, Identification and Authentication. This section describes the policies applied by each mechanism that creates user sessions. User attributes for identification and authentication are described in section 6.2.2, User Attributes.

7.1.1 Interactive Login and Passwords

A user logs in to the RS/6000 Distributed System using the console of any host in the distributed system. Passwords are used for user authentication. Once a user has successfully logged in to one host in the RS/6000 Distributed System, that user can access other hosts in the distributed system. Some services, such as accessing files on NFS-mounted file systems do not require the user to log in again. Other services, such as using telnet to obtain a shell prompt on another host or using ftp to transfer files between hosts in the distributed system require logging in again in order to enforce individual accountability.

As noted in section 5.3.10, Identification and Authentication, the contents of */etc/security* (and other configurable data) are stored on one host in the distributed system, and are NFS-mounted on all other hosts. Thus, the password information available to all hosts in the distributed system is identical and configuration parameters changed on one host are propagated throughout the distributed system.

7.1.1.1 Administrative Configuration Options Effecting TCB Policies

AIX provides a number of configurable options that bear directly on administration of the security policy. These options are in the TCB database configuration file, */etc/security/user*. These are per-user attributes with defaults defined for each that are applied system wide and may be overridden on a per-user basis, as shown in Table 7.1.

Table 7.1. Administrative Configuration Options. *AIX provides a number of per-user attributes.*

Option	Description
account_locked	Indicates if a user allowed access to the system
dictionlist	Password dictionaries to be consulted when new passwords are generated.
expires	Identifies the expiration date of the account.
histexpire	Defines the period of time that a user cannot reuse a password.
histsize	Defines the number of previous passwords that a user cannot reuse.
login	Defines whether or not a user can log in using the login command.
logintimes	Specifies the days and times of day when a user is permitted to login.
loginretries	Defines the number of unsuccessful login attempts allowed before locking the account.
maxage	Defines the maximum lifetime of a password.
maxexpired	Defines the grace period beyond maxage when a user may change a password without administrative intervention.
maxrepeats	Defines the maximum number of times a character can be repeated in a new password.
minalpha	Defines the minimum number of alphabetic characters that must be in a new password.
mindiff	Defines the minimum number of characters in a new password that must not have been in the previous password.
minlen	Defines the minimum length of a password.
minother	Defines the minimum number of non-alphabetic characters required in a new password.
rlogin	Permits access to the account using either telnet or rlogin.
su	Indicates whether or not a user may change identities to this user.
sugroups	Lists the groups from which a user may change to this user identity.
ttys	Lists the terminals that can access the account or should be denied access to the account. rsh and rexec can be permitted or denied using this.
umask	Determines default file permissions.

7.1.1.2 Password Authentication

The standard authentication mechanism in the RS/6000 Distributed System is passwords. There is a configuration user attribute for minimum password length. Passwords may be forced to expire periodically. If the user's password has expired, that user is not permitted to log in until the user or administrator has changed the password.

AIX users are permitted to change their passwords. Users choose their own passwords based on guidance in the SFUG, which advises users to choose passwords that are not derived from personal or other information that can be easily guessed by an attacker. AIX does not include an automatic password generation facility. An administrator can configure password quality parameters that must be enforced. Each of these user attribute can be set on a system-wide basis or for an individual user. If the same user attribute is set for both the default and specific user case, the user-specific value overrides the default value. Only the password quality user attribute for minimum length (minlen) must be used in the evaluated configuration, and it must be set to eight (8). There are other password quality user attribute, but they are not required to meet the C2 level of trust requirements but are highly recommended for use as they do enhance the system security posture. The TFM contains definitions for these other user attribute, if a system administrator desires to enforce them.

7.1.1.3 Changing Identity Policy

Users can change their identity (*i.e.*, switch to another identity) using the su command. When switching identities, the login UID is not changed, so all actions are ultimately traceable in the audit trail to the originating user. For purposes of this discussion, the original UID is called the source identity and the UID being switched to is the target identity.

In order for a user to su to a different user identity, all of the following must be true:

- The entry in */etc/security/user* for the target identity must have the su user attribute set to true.
- The entry in */etc/security/user* for the target identity must have the sugroups user attribute set to allow at least one group that the source identity is a member of, and sugroups must not deny any groups that the source identity is a member of.
- The user must present the correct password corresponding to the target identity. The password is hashed and compared to the stored hashed password in */etc/security/passwd* for verification.

The TFM requires that the sugroups user attribute for the root user identity, and all other administrative user accounts, be set to allow only those users that are designated as administrators to change to that identity. This prevents non-administrators from logging in under their own identity and then attacking administrator accounts. An administrator must login with a non-root identity, then su to the root identity to enforce individual accountability.

7.1.1.4 Authentication Failure Handling

Administrators can configure login parameters to define the number of allowable unsuccessful login attempts before the system locks the account. An administrator can utilize login configuration parameters to define limits for such things as:

- the delay between unsuccessful login attempts
- the number of unsuccessful attempts before an account is locked
- the number of seconds during which unsuccessful login attempts must occur before an account is locked
- the number of minutes after an account is locked that it can be automatically unlocked (or if set to 0, the account remains locked until the administrator unlocks it).

An account can be automatically disabled (and either manually or automatically re-enabled) after a set number of incorrect password guesses. Users will also be unable to login if their account has been retired by the administrator.

7.1.2 Batch Authentication

Batch job authentication is subject to the same authentication mechanisms as a regular login. Only authorized users can submit and access batch jobs. When a batch job is considered for execution, the UID is checked to ensure that it is a valid UID (i.e. has not been deleted) and, if it is valid, the job will execute under the UID, and the associated process attributes of that UID. Batch jobs are

associated with the real UID of the process submitting the request. If a user has switched identities to another user using the *su* command, and submits a batch job, the batch job will run under the identity of the target user.

As explained in section 5.3.12, Batch Processing, there are two types of files that determine which users are authorized to submit (and modify) their own batch jobs: *allow* and *deny*.

There is only one access mode for batch jobs. The access mode allows the user complete access to their own job. In addition, batch jobs with an effective UID of 0, or the root identity, have complete access to all queues. Group IDs do not play a role in access controls for batch jobs.

When a user submits a batch job, the user's UID, GID, supplementary groups, login ID, and audit information are recorded. These values are used when the batch job is run, thus ensuring that audit information is associated with the user whose session originated the request, while using the permissions associated with the real UID at the time the batch job was submitted.

7.2 Privileges

There are two modes of privilege available in AIX 4.3, least privilege mode and monolithic mode. Both modes use four privilege vectors: inherited, effective, bequeathed, and maximum. These privilege vectors are stored as part of the process's credential structure.

In the evaluated system, the monolithic mode is the only supported privilege mode. There is a system configuration parameter that is set to ensure that least privilege mode is not enabled.

7.2.1 Process Privilege Sets

In monolithic privilege mode if the effective UID of the process is root, all privileges are enabled in all privilege vectors. If the process is not root, there are no privileges enabled in any of the privilege vectors.

7.2.1.1 fork

If a process performs a ***fork*** operation the resulting process will have the same privileges as the parent, i.e. if root ***forks*** a child process the child has all privileges, and if an untrusted process ***forks*** a child process the child has no privileges.

7.2.1.2 exec

In monolithic privilege mode, the privilege vectors are set based on the effective user identity at the time the executable image is started via ***exec***. Processes which have an effective user identity of root have all the bits enabled, and processes which have an effective user identity other than root have none of the bits enabled.

7.2.1.3 setuid

When a process changes identity, such as using one of the setuid system calls, all of the privilege vectors are automatically recalculated to reflect the process's new identity.

7.2.2 Privilege Control Lists

Privilege control lists (PCLs), which are associated with executable files, contain a set of additional privileges to be inherited by a process and a list of user and/or group identities which are to inherit those privileges. PCLs, when they exist, are stored in the file's extended inode.

Since the use of PCLs cannot be restricted by the least privilege system parameter, there are no TCB files delivered with PCLs. The evaluated configuration does not contain the commands that allow setting or manipulating of the PCLs and the administrator is instructed in the TFM to not set PCLs on any TCB files. The command **tcbck** can also be used to detect any unauthorized PCLs.

The above controls completely restrict the use of PCLs in the evaluated configuration.

7.3 Discretionary Access Control

This section outlines the general DAC policy in the RS/6000 Distributed System as implemented for resources where access is controlled by permission bits and optionally, extended permissions; principally these are the objects in the file system. In all cases the policy is based on user identity (and in some cases on group membership associated with the user identity). To allow for enforcement of the DAC policy, all users must be identified and their identities authenticated. Details of the specific DAC policy applied to each type of resource are covered in section 7.4, Discretionary Access Control: File System Objects and section 7.5, Discretionary Access Control: IPC Objects.

The general policy enforced is that subjects (*i.e.*, processes) are allowed only the accesses specified by the class-specific policies. Further, the ability to propagate access permissions is limited to those subjects who have that permission, as determined by the class-specific policies. Finally, a subject with an effective UID of 0 is exempt from all restrictions and can perform any action desired.

DAC provides the mechanism that allows users to specify and control access to objects that they own. DAC attributes are assigned to objects at creation time and remain in effect until the object is destroyed or the object attributes are changed. DAC attributes exist for, and are particular to, each type of object on the RS/6000 Distributed System. DAC is implemented with permission bits and, when specified, extended permissions.

A subject whose effective UID matches the file owner ID can change the file attributes, the base permissions, and the extended permissions. Changes to the file group are restricted to the owner. The new file group identifier must either be the current effective group identifier or one of the group identifiers in the concurrent group set. In addition, a subject whose effective UID is 0 can

make any desired changes to the file attributes, the base permissions, the extended permissions, and owning user of the file.

Permission bits are the standard UNIX DAC mechanism and are used on all RS/6000 Distributed System file system named objects. Individual bits are used to indicate permission for read, write, and execute access for the object's owner, the object's group, and all other users (i.e. world). The extended permission mechanism is supported only for file system objects and provides a finer level of granularity than do permission bits.

7.3.1 Permission Bits

The RS/6000 Distributed System uses standard UNIX permission bits to provide one form of DAC for file system named objects. There are three sets of three bits that define access for three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w) and one for execute (x). Each subject's access to an object is defined by some combination of these bits:

- rwx symbolizing read/write/execute
- r-x symbolizing read/execute
- r-- symbolizing read
- --- symbolizing null

When a process attempts to reference an object protected only by permission bits, the access is determined as follows:

- Effective UID = Object's owning UID and the owning user permission bits allow the type of access requested. Access is granted with no further checks.
- Effective GID, or any supplementary groups of the process = Object's owning GID, and the owning group permission bits allow the type of access requested. Access is granted with no further checks.
- If the process is neither the owner nor a member of an appropriate group and the permission bits for world allow the type of access requested, then the subject is permitted access.
- If none of the conditions above are satisfied, and the process is not the root identity**,** then the access attempt is denied.

7.3.2 Extended Permissions

The extended permissions consist of an essentially unlimited number of additional permissions and restrictions for specific users and groups. Each entry in the extended permissions list consists of three parts: an entry type, a set of permissions, and an identity list.

- The entry type is the value permit, deny, or specify (indicating that the entry indicates a set of permissions to be allowed as supplemental to the listed identity(ies), denied to the listed identity(ies), or that the permissions permitted and the complementary set denied to the listed identity(ies) respectively).
- The permission set is zero or more of the permissions read, write, and execute.
- The identity list is one or more values specifying users and/or groups. The entry is applied if the process' effective UID, effective GID, and supplemental groups match all values in the list. The term "match" means that for each value in the identity list, if the value is for a UID, that the specified UID is the same as the process' effective UID, and if the value is for a GID, that the specified GID is either the same as the process' effective GID or the specified GID is included in the process' list of supplemental GIDs.

There is no explicit ordering of entries within the extended permissions. To determine access rights, the kernel takes into account all entries that match the UID or GID of the process. For each entry, the permit and specify bits are added to a permissions list and the deny and bitwise negation of the specify are added to a restrictions list. The restrictions are bitwise removed from the permissions and the resulting list is used in the access determination.

The maximum size for the extended permissions is one memory page (4096 bytes). The entries are variable length. Each entry takes a minimum of 12 bytes (two for the length of the entry, two for the permission type and permissions allowed, two for the number of identity entries, two for the type of identity entry, and four for each UID/GID). As a result, there can be over 300 entries in an extended permissions list, which is in practice unlimited.

Collectively, the file attributes, base permissions, extended permissions, and extended attributes are known as the file Access Control List (ACL). ACLs have a textual representation (used with commands such as ACLGET) and binary representations (for storage in the file system). Table 7-2 is a textual representation of an ACL.

When a process attempts to reference an object protected by an ACL, it does so through a system call (e.g., ***open, exec***). If the object has been assigned an ACL access is determined as according to the algorithm below:

Table 7-2. Sample File System ACL.

ACL Definition	Explanation
attributes: SUID	Specifies that the setuid bit is turned on and the setgid and savetext bits are turned off.
base permissions:	This line is optional, and indicates that the following lines are defining the base permissions.
owner(frank): rw-	The owning user of the file, who is "frank", has the *read* and *write* permissions to the file.
group(system): r-x	The owning group of the file, who is "system", has the *read* and *execute* permissions to the file.
others: ---	Those individuals who are neither the file owner nor file group do not get any permissions to from the base permissions (but may obtain permissions from the extended permissions).
extended permissions:	This line is optional, and indicates that the following lines are defining the extended permissions.
enabled	Specifies that extended permissions are enabled (if this line specified ***disabled***, then the rest of the ACL would be ignored.)
permit rw- u:sally	Grants user (u:) "sally" the ***read*** and ***write*** permissions to the file.
deny r-- u:nicole, g:system	Denies user (u:) "nicole" the *read* permission to the file when she is a member of the "system" group.
specify r-- u:gary, g:projx, g:projy	Grants user "gary" the read permission, and denies both write and execute permission, so long as he is a member of both the "projx" and "projy" groups. If "gary" is not a member of both groups, this extended permission entry is ignored.
permit rw- g:account, g:finance	Grants the ***read*** and ***write*** permissions to any user who is a member of both the "account" and "finance" groups.

7.3.3 Pathname Traversal and Access Decision Flowchart

The flowchart in figure 7.1 illustrates the process that is used to resolve an AIX pathname and determine if a subject has access to a named file system object. The input to this algorithm is the subject (and in particular the credentials of the subject), the object (represented by a pathname string) and the access mode.

This algorithm makes use of several terms for describing the pathname resolution process. The access directory is the current point in the pathname resolution process. The current object is the result of searching the access directory and is either the final requested object or the next object to be used as the access directory.

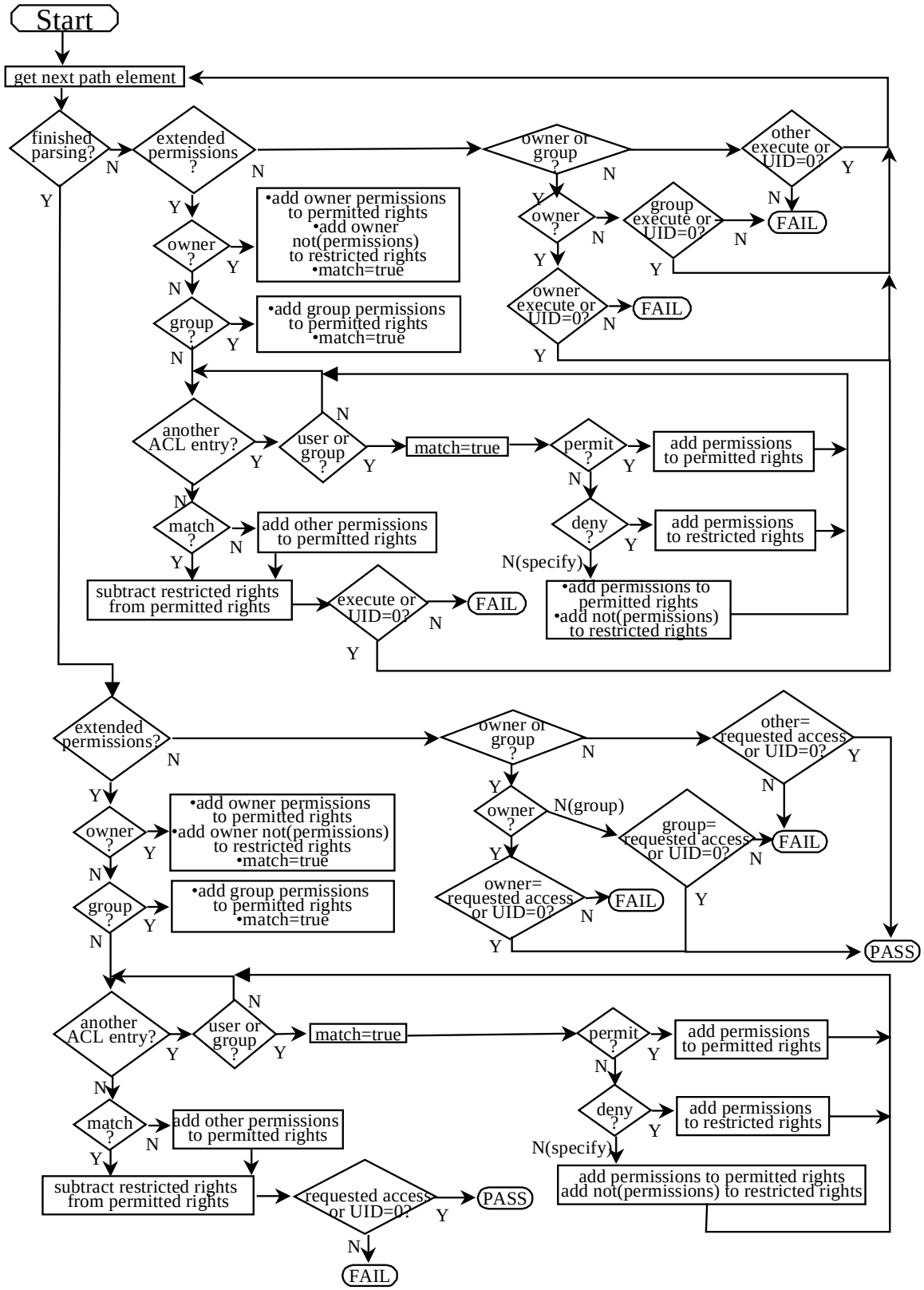

Figure 7.1. Pathname Traversal and Access Decision Flowchart. *A subject must have search permission for every element of the pathname and the requested access for the object.*

7.4 Discretionary Access Control: File System Objects

The Discretionary Access Control (DAC) policy is described above. This section describes the details of DAC policies as they apply to file system objects.

7.4.1 Common File System Access Control

This section describes the common DAC policy applied to file system objects, including policies for object contents and attributes.

7.4.1.1 DAC Contents Policy

The permission-bit and ACL DAC policy determines the effective access that a process may have to the contents of a file system object: some combination of read(r), write (w), and execute (x). In general, read access permits the object's contents to be read by a process, and write permits them to be written; execute is interpreted differently for different object types. Some object types (unnamed pipes, symbolic links) do not use the permission bits at all.

7.4.1.2 DAC Attributes Policy

In general, a process must be the object's owner, or have privilege, to change the objects attributes, and there are no DAC restrictions on viewing the attributes, so any process may view them. However, the following are exceptions to the rule:

- The permission bits and ACL (permission bits, extended permissions and attributes) of an object may be changed by an owner or by the root identity.
- The owning group ID of an object may be changed by an owner, but only to a group of which the process is currently a member, unless it is the root identity.
- The owning user ID of an object may only be changed by the root identity.

7.4.1.3 DAC Defaults

The default access control on newly created FSOs is determined by the permissions associated with the directory where the FSO was created, the effective user ID, group ID, and umask value of the process that created the FSO, and the specific permissions requested by the program creating the FSO.

- The owning user of a newly created FSO will be the effective UID of the creating process.
- If the setgid bit is set on the containing directory, then the owning group of a newly created FSO will be the owning group of the containing directory. If the setgid bit is not set on the containing directory, then the owning group of the newly created FSO will be the effective GID of the creating process.

- The initial access permissions on the FSO are those specified by the creating process bit-wise ANDed with the one's complement of the umask value. For example, if a program specified initial permissions of 0664 (read/write for owner, read/write for group, and read for world) but the umask value were set to 0027 (prevent write for group or world, prevent all permissions for world), then the initial file permissions would be set to 0644 (or 0644 bit-and 0750).
- There are initially no extended permissions associated with an FSO. Extended permissions can be set by applications or by users using AIX commands.

Base and extended access permissions can be changed by any process with an effective UID equal to the owning UID of the FSO, providing that the effective UID has at least the execute permission to the containing directory. Note that since a file may have multiple hard links, the process can use any of the containing directories (*i.e.*, if there is any directory containing a link to the file, then that path could be used as a means to get to the file and change its permissions).

7.4.1.4 DAC Revocation on File System Objects

With the exception of NFS objects, file system objects (FSOs) access checks are performed when the FSO is initially opened, and are not checked on each subsequent access. Changes to access controls (*i.e.*, revocation) are effective with the next attempt to open the FSO.

For NFS objects, access is checked for each operation. A change to the access rights for an NFS FSO take effect as of the next NFS request.

In cases where the administrator determines that immediate revocation of access to an FSO is required, the administrator can reboot the computer, resulting in a close on the FSO and forcing an open of the FSO on system reboot. This method is described in the TFM.

7.4.2 DAC: Ordinary File

In addition to read and write, ordinary files support the execute permission. Execute access is required to execute the file as a program or script. When an executable file has the set-user-ID or set-group-ID flags set, or is executed by the root identity, executing it as a program changes the process's security attributes.

7.4.3 DAC: Directory

The execute permission bit for directories governs the ability to name the directory as part of a pathname. A process must have execute access in order to traverse the directory during pathname resolution.

Directories may not be written directly, but only by creating, renaming, and removing (unlinking) objects within them. These operations are considered ***writes*** for the purpose of the DAC policy.

7.4.4 DAC: Device Special File

The access control scheme described for FSOs is used for protection of character and block device special files. Most device special files are configured to allow read and write access by the root user, and read access by privileged groups. With the exception of terminal and pseudo-terminal devices and a few special cases (*e.g.*, */dev/null* and */dev/tty*), devices are not accessible to non-TCB software.

7.4.5 DAC: UNIX Domain Socket Special File

UNIX domain socket files are treated identically to any other file in the AIX file system from the perspective of access control, with the exception that using the ***bind*** or ***connect*** system calls requires that the calling process must have both read and write access to the socket file.

Because UNIX domain sockets exist in the file system name space, the socket files can have both base mode bits and extended ACL entries. Both are considered in making the DAC decision.

UNIX domain sockets consist of a socket special file (managed by the Logical File System) and a corresponding socket structure (managed by IPC). The LFS controls access to the socket based upon the caller's rights to the socket special file.

7.4.6 DAC: Named Pipes

Named pipes are treated identically to any other file in the AIX file system from the perspective of access control.

7.4.7 DAC: Special Cases for NFS File Systems

Only files, directories, and symbolic links have accessible contents in the NFS file system, and only those object types may be created. However, those attributes supported by NFS can be manipulated regardless of object type.

DAC checks by the NFS server for file contents permit a subject with the same effective owning user ID as the file to have access to the contents regardless of the DAC attributes. This is used to support the standard UNIX semantics for access to open files, because such access is not re-validated when a file's DAC attributes change. This special case relies on the property that, ordinarily, only a file's owner changes its DAC while the file is open, and it is thus sufficient to handle the owner specially.

DAC changes do have immediate effect for users other than the owner, unlike local files: if an NFS-accessed file's DAC is changed to deny access, any subsequent read or write operation to an open file will fail if the operation would no longer be permitted by the new DAC attributes. However, this can never grant additional access, because the client would have checked the access when the file was opened and not permitted more access than the DAC attributes allowed at open time.

In addition these access checks are always based on the process's current identity, rather than the identity at the time the file was opened as is done in some UNIX systems. Thus a set UID program that opens an NFS-mounted file and subsequently changes its user ID back to the saved user ID may lose access to the file.

7.5 Discretionary Access Control: IPC Objects

7.5.1 DAC: Shared Memory

For shared memory segment objects (henceforth SMSs), access checks are performed when the SMS is initially attached, and are not checked on each subsequent access. Changes to access controls (*i.e.*, revocation) are effective with the next attempt to attach to the SMS.

In cases where the administrator determines that immediate revocation of access to a SMS is required, the administrator can reboot the computer, thus destroying the SMS and all access to it. This method is the described in the TFM. Since a SMS exists only within a single host in the distributed system, rebooting the particular host where the SMS is present is sufficient to revoke all access to that SMS.

If a process requests deletion of a SMS, it is not deleted until the last process that is attached to the SMS detaches itself (or equivalently, the last process attached to the SMS terminates). However, once a SMS has been marked as deleted, additional processes cannot attach to the SMS and it cannot be undeleted.

The default access control on newly created SMSs is determined by the effective user ID and group ID of the process that created the SMS and the specific permissions requested by the process creating the SMS.

- The owning user and creating user of a newly created SMS will be the effective UID of the creating process.
- The owning group and creating group of a newly created SMS will be the effective GID of the creating process.
- The creating process must specify the initial access permissions on the SMS, or they are set to null and the object is inaccessible until the owner sets them.
- SMSs do not have extended permissions.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the SMS. Access permissions can also be changed by any process with an effective UID of 0, also known as running with the root identity.

7.5.2 DAC: Message Queues

For message queues, access checks are performed for each access request (*e.g.*, to send or receive a message in the queue). Changes to access controls (*i.e.*, revocation) are effective upon the next request for access. That is, the change affects all future send and receive operations, except if a process has already made a request for the message queue and is waiting for its availability (*e.g.*, a

process is waiting to receive a message), in which case the access change is not effective for that process until the next request.

If a process requests deletion of a message queue, it is not deleted until the last process that is waiting for the message queue receives its message (or equivalently, the last process waiting for a message in the queue terminates). However, once a message queue has been marked as deleted, additional processes cannot perform messaging operations and it cannot be undeleted.

The default access control on newly created message queues is determined by the effective user ID and group ID of the process that created the message queue and the specific permissions requested by the process creating the message queue.

- The owning user and creating user of a newly created message queue will be the effective UID of the creating process.
- The owning group and creating group of a newly created message queue will be the effective GID of the creating process.
- The initial access permissions on the message queue must be specified by the creating process, or they are set to null and the object is inaccessible until the owner sets them.
- Message queues do not have extended permissions.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the message queue. Access permissions can also be changed by any process with an effective UID of 0.

7.5.3 DAC: Semaphores

For semaphores, access checks are performed for each access request (*e.g.*, to lock or unlock the semaphore). Changes to access controls (*i.e.*, revocation) are effective upon the next request for access. That is, the change affects all future semaphore operations, except if a process has already made a request for the semaphore and is waiting for its availability, in which case the access change is not effective for that process until the next request.

In cases where the administrator determines that immediate revocation of access to a semaphore is required, the administrator can reboot the computer, thus destroying the semaphore and any processes waiting for it. This method is the described in the TFM. Since a semaphore exists only within a single host in the distributed system, rebooting the particular host where the semaphores is present is sufficient to revoke all access to that semaphore.

If a process requests deletion of a semaphore, it is not deleted until the last process that is waiting for the semaphore obtains its lock (or equivalently, the last process waiting for the semaphore terminates). However, once a semaphore has been marked as deleted, additional processes cannot perform semaphore operations and it cannot be undeleted.

The default access control on newly created semaphores is determined by the effective user ID and group ID of the process that created the semaphore and the specific permissions requested by the process creating the semaphore.

- The owning user and creating user of a newly created semaphore will be the effective UID of the creating process.
- The owning group and creating group of a newly created semaphore will be the effective GID of the creating process.
- The initial access permissions on the semaphore must be specified by the creating process, or they are set to null and the object is inaccessible until the owner sets them.
- Semaphores do not have extended permissions.

Access permissions can be changed by any process with an effective UID equal to the owning UID or creating UID of the semaphore. Access permissions can also be changed by any process with an effective UID of 0.

7.6 Object Reuse

Object Reuse is the mechanism that protects against *scavenging*, or being able to read information that is left over from a previous subject's actions. Three general techniques are applied to meet this requirement in the RS/6000 Distributed System: explicit initialization of resources on initial allocation or creation, explicit clearing of resources on release or deallocation, and management of storage for resources that grow dynamically.

Explicit initialization is appropriate for most TCB-managed abstractions, where the resource is implemented by some TCB internal data structure whose contents are not visible outside the TCB: queues, datagrams, pipes, and devices. These resources are completely initialized when created, and have no information contents remaining.

Explicit clearing is used in the RS/6000 Distributed System only for directory entries, because they are accessible in two ways: through TCB interfaces both for managing directories and for reading files. Because this exposes the internal structure of the resource, it must be explicitly cleared on release to prevent the internal state from remaining visible.

Storage management is used in conjunction with explicit initialization for object reuse on files, and processes. This technique keeps track of how storage is used, and whether it can safely be made available to a subject.

7.6.1 Object Reuse: File System Objects

All file system objects (FSOs) available to general users are accessed by a common mechanism for allocating disk storage and a common mechanism for paging data to and from disk. This includes the Journaled File System (JFS) and Network File System (which exists physically as a JFS volume on a server host). It includes both normal and large JFS file systems.

Object reuse is irrelevant for the CD-ROM File System (CDRFS) because it is a read-only file system and so it is not possible for a user to read residual data left by a previous user. File systems

on other media (tapes, diskettes.) are irrelevant because of TFM warnings not to mount file systems on these devices.

For this analysis, the term FSO refers not only to named file system objects (files, directories, device special files, named pipes, and UNIX domain sockets) but also to unnamed abstractions that use file system storage (symbolic links and unnamed pipes). All of these, except unnamed pipes, have a directory entry that contains the pathname and an inode that controls access rights and points to the disk blocks used by the FSO.

In general, file system objects are created with no contents, directories and symbolic links are exceptions, and their contents are fully specified at creation time.

7.6.1.1 Object Reuse: Files

Storage for files is allocated automatically in pages as a file grows. These pages are cleared before they become accessible, within the file. However, when a file is deleted the space holding the data from the file, both in memory and on disk, is not cleared. This data will persist until the space is assigned to another file, when it will be cleared. These *internal fragments* of deleted files are protected by the kernel to prevent accessing of deleted data.

If data is read before it is written, it will read only as zeroes. Reads terminate when the end-of-file (EOF) is detected. It is possible to seek past the EOF, but any reads will return zeros. File writes may cause the file to grow, thus overwriting any residual data and moving the EOF. If the file is seeked past the EOF and then written, this leaves a hole in the file that will subsequently be read as zeroes.

7.6.1.2 Object Reuse: Directories and Directory Entries

In part, object reuse for directories is handled as for ordinary files: pages allocated are always cleared before being incorporated into the directory. When a directory is first created, it is explicitly initialized to have the entries "." and "..", but the remainder of the directory's storage is cleared.

Individual directory entries are manipulated as distinct resources, such as when referencing file system objects, and as part of the directory, such as when reading the entire directory itself. When a directory entry is removed or renamed the space occupied by that directory entry is either combined with the previous entry as free space or else the i-node number of the entry is set to zero when the entry occurs on a 512 byte boundary.

When a directory entry does not occur on a 512-byte boundary, the size of the preceding directory entry in incremented by the size of the directory entry which has been removed. The space in a directory entry in excess of that which is needed to store the necessary information may be allocated when a directory entry is to be created. The fields of the directory entry remain unchanged.

When a directory entry occurs on a 512-byte boundary, the i-node number is set to zero to indicate that this entry is now available for re-use. All other fields of the directory entry remain unchanged.

The directory entry is no longer visible to interfaces which perform file name operations and may only be seen when the entire directory is examined and the process has read access to the directory.

7.6.1.3 Object Reuse: Symbolic Links

Symbolic links have their contents (the link pathname) fully specified at creation time, and the readlink operation returns only the string specified at creation time, not the entire contents of the block it occupies.

7.6.1.4 Object Reuse: Device Special Files

All device special files are initialized to a known state on first open and never grow.

7.6.1.5 Object Reuse: Named Pipes

FIFOs are created empty. Buffers are allocated to contain data written to a pipe, but the read and write pointers are managed to ensure that only data that was written to the pipe can ever be read from it.

7.6.1.6 Object Reuse: Unnamed Pipes

Unnamed pipes are created empty. Buffers are allocated to contain data written to a pipe, but the read and write pointers are managed to ensure that only data that was written to the pipe can ever be read from it.

7.6.1.7 Object Reuse: Socket Special File (UNIX Domain)

UNIX domain sockets have no contents; they are fully initialized at creation time.

7.6.2 Object Reuse: IPC Objects

System V shared memory, message queues, and semaphores are initialized to all zeroes at creation. These objects are of a finite size (shared memory segment is from one byte to 256 MBytes, semaphore is one bit), and so there is no way to grow the object beyond its initial size. No processing is performed when the objects are accessed or when the objects are released back to the pool.

7.6.3 Object Reuse: Queuing System Objects

7.6.3.1 Object Reuse: Printer Job Description Files

A print queue consists of a sequence of job description files. When the queue daemon starts up, it reads the job description files to establish its in-memory representation of the queue. This queue is used to dispatch jobs, to respond to user status requests, and to handle requests to delete queue entries. Object reuse for the job description files are handled as for files as described previously.

7.6.3.2 Object Reuse: Batch Queue Entries

cron and ***at*** jobs are defined in batch files, which are subject to the object reuse protections specified for files as described previously.

7.6.4 Object Reuse: Miscellaneous Objects

7.6.4.1 Object Reuse: Process

A new process's context is completely initialized from the process's parent when the ***fork*** system call is issued. All program visible aspects of the process context are fully initialized. All kernel data structures associated with the new process are copied from the parent process, then modified to describe the new process, and are fully initialized.

The AIX kernel zeroes each memory page before allocating it to a process. This pertains to memory in the program's data segment and memory in shared memory segments. When a process requests more memory, the memory is explicitly cleared before the process can gain access to it.

When the kernel performs a context switch from one thread to another, it saves the previous thread's General Purpose Registers (GPRs) and restores the new thread's GPRs, completely overwriting any residual data left in the previous thread's registers. Floating Point Registers (FPRs) are saved only if a process has used them. The act of accessing an FPR causes the kernel to subsequently save and restore all the FPRs for the process, thus overwriting any residual data in those registers.

Processes are created with all attributes taken from the parent. The process inherits its memory (text and data segments), registers, and file descriptors from its parent. When a process execs a new program, the text segment is replaced entirely. The AIX kernel zeroes each memory pages before allocating it to a process. This pertains to memory in the program's data segment and memory in shared memory segments (shmat, mmap). Each thread is associated with a process and has access to all of the process resources. The allocation and deallocation of a thread affects only its process. When the kernel performs a context switch from one thread to another, it saves the previous thread's General Purpose Registers (GPRs) and restores the new thread's GPRs, thus, completely overwriting any residual data left in the previous thread's registers.

Floating point registers (FPRs) are saved only if they have been used by a process. However, the act of accessing a FPR causes the kernel to subsequently save and restore all the FPRs for the process, thus overwriting any residual data in these registers.

7.6.4.2 Object Reuse: Datagrams

The buffers that are allocated to hold datagrams control information are zeroed out on allocation. The buffers that are allocated to hold network data are not zeroed out because the user data or inbound packet from the network entirely overwrites the data in the buffer.

7.6.4.3 Object Reuse: Mail Files

Mail spool files are files managed by the **SENDMAIL** program. Residual data for these files is handled through the underlying mechanisms for file system objects.

7.6.4.4 Object Reuse: Printer DRAM

Every print job contains a reset command at the beginning and end of the print job. The reset command removes all user data and temporary macros from the printer DRAM, initializing the printer DRAM to a known empty state. Permanent macros remain. In addition to the reset command, an escape sequence that removes all macros, temporary and permanent is also sent prior to the print job executing.

7.6.4.5 X Windows Resources and Frame Buffer

The X Windows system permits user software to directly access the frame buffer and registers on the video adapter. The Rendering Context Manager (RCM) kernel software ensures that the user that started the X Server has exclusive access to these hardware resources. Because users have serially exclusive access to the video hardware, the X Server and video adapter can be analyzed as an aggregate storage object that must be relinquished by the current user before a subsequent user can access the object.

When a user terminates an X-windows session the file descriptor associated with the graphics devices is discarded and control of the graphics adapter is returned to the LFT. Upon this return, the LFT reinitializes the video adapter hardware to a clear state.

7.7 Audit

7.7.1 Summary of Audit Events

A user account is audited based on the classes that have been assigned to it. A description of audit classes and the process of assigning them to a user may be found in section 5.3.15.2, Audit Control.

Audit classes are administrator configurable aggregations of audit events. The system-defined classes are shown in table 7-3 below. An administrator may add or delete classes or change the system-defined classes as they desire, with the exception of the "ALL" class, which has an implicit definition.

A complete list of auditable events is located in Appendix G.

Table 7-3. Audit Classes. *Audit classes split the auditable events into smaller groups.*

Class	Description
general	User id changes, password changes, creation or deletion of directories, changing directories or changing the root directory
objects	Modification to any of a group of system configuration files
SRC	Starting and stopping of daemons and subsystems
kernel	Process operations including create, delete and execute
files	File operations including open, read, write, close, rename, change permissions, change owner and change mode
svipc	Message queues, semaphore and shared memory events
mail	Modification to the **SENDMAIL** configuration files
cron	**AT** and **CRON** job operations
tcpip	Changes to TCP/IP configuration, connections, data in and data out
lvm	Logical volume manager operations, including changes to logical and physical volumes
ALL	All defined audit events

7.7.2 Audit: File System Objects

There are two methods of auditing file system objects. The first method occurs when a user attempts to access a file. The */etc/security/audit/objects* file contains a list of the objects that are audited on the system, based on the object's access rights. The second method occurs when a user's process attempts to perform an operation on a file, such as creating or opening a file.

A list of the file system object events that are provided by the RS/6000 Distributed System is located in Appendix G. These events are differentiated from normal file system events by the word 'FILE' at the beginning of the event name.

7.7.3 Audit: Device and Media Resources

7.7.3.1 Audit: Tape Resources

Backup and restore operations are audited when an export or import operation happens for the system.

A list of the backup and restore events provided by the RS/6000 Distributed System is located in Appendix G. These events are located at the bottom of the Logical Volume Manager table.

7.7.3.2 Audit: Printer Resources

Printer requests that are queued using the ENQ command are auditable, as well as an action performed by QDAEMON in the printing process.

A list of the printing events provided by the RS/6000 Distributed System is located in Appendix G. These events are located in the command table.

7.7.3.3 Audit: File Systems

The file system events available for auditing are: ***mount*** and ***unmount*** of a file system, extend the size of a file system, change directory, make directory, remove directory and change root.

A list of the file system events provided by the RS/6000 Distributed System is located in Appendix G. These events are differentiated from the file system object events by the letters 'FS' at the beginning of the event name.

7.7.4 Audit: Deferred Execution Resources

Deferred execution resources include the AT and CRON facilities. Two audit events are provided for the AT method of delayed execution: add a job and remove a job. Four audit events are provided for the CRON method of delayed execution: add a job, remove a job, start of job and finish of job. Jobs added through the at facility are executed by the CRON facility and will be audited using the events for the CRON facility's start of job and finish of job

A list of deferred execution resource events provided by the RS/6000 Distributed System is located in Appendix G. These events are located in the commands table.

7.7.5 Audit: Network Resources

A collection of events is available to audit TCP/IP related actions on the system. These events include the ability to audit the establishment and/or closure of a TCP/IP connection, and the transfer of data using TCP/IP applications like ftp.

A list of TCP/IP audit events provided by the RS/6000 Distributed System is located in Appendix G.

8. ASSURANCES

8.1 System Architecture

The RS/6000 Distributed System implements a two-state architecture, where kernel mode software runs with hardware privilege (Supervisor State) and user mode software runs without hardware privilege (Problem State). In kernel mode, processes exist only in the kernel protection domain and run with hardware privilege. In user mode, a process executes application code while the machine is in a non-privileged state. The kernel data and global data structures are protected through the employment of the hardware protection mechanisms as described in Chapter 4 (TCB Hardware).

8.2 System Integrity Tests

The TCSEC requires the capability for periodically validating the correct operation of the TCB hardware and firmware. For the RS/6000 Distributed System, this is addressed on each individual computer by Power On Self Test (POST) firmware in non-volatile memory and by diagnostic software for the various RS/6000 Distributed System devices.

8.2.1 Power On Self Test (POST)

Each of the three models (43P, F50, and S70) includes Power On Self-Test (POST) firmware in non-volatile memory. The code is model dependent because the hardware to be tested is different for each computer. The POST runs when computers are powered-on or reset and tests the following capabilities:

- registers read and write
- setable bits are set and reset (stuck fault testing)
- level 2 cache read and write (pattern testing)
- memory read and write (pattern testing)
- device controllers read and write(e.g., keyboard, serial port)

The 43P does not have a separate service processor. However, F50 and the S70 have a separate service processor. The POST test for the S70 performs the following:

- hardware wire tests (e.g., solder joints)
- Built-in Self Test
- service processor test

8.2.2 Diagnostic Software

AIX provides diagnostics for the resources of the evaluated hardware. Diagnostics are provided for the PCI bus, SCSI controller, tape drive, hard drive, CD ROM, graphics adapter, network adapters (Ethernet and token ring), ISA Bus, keyboard, mouse, floppy, and parallel port. These

diagnostics are included as part of the devices LPP. The diagnostics can only run in service mode (prior to multi-user mode). Although different diagnostics exist for each of these resources, they all test the following capabilities for all levels supported by the hardware:

- interrupts
- DMA (read and write)
- configuration register read and write
- I/O register read and write
- read/write capability
- input/output capability
- write protect capability
- adapter initialization
- internal loopback (transmit and receive)
- external loopback (transmit and receive)

The administrator is notified if any error is detected while running the diagnostics. In addition, the administrator may capture and review the results of these tests in error logs.

8.3 Design Documentation

IBM produced mid-level design documentation according to the PAT Guidance Working Group Form and Content of Vendor Design Documentation. The Architecture Summary Document (ASD) and the Interface Summary Document (ISD) provided an up-to-date, accurate mid-level description of the system architecture and it's interfaces. Information on lower-level design and implementation was provided through additional IBM documentation, publicly available documentation, interviews with developers, internal training, and code-inspections. A complete list of all the publicly available documentation referenced in the course of this evaluation can be found in Appendix E: Bibliography and References.

8.4 User Documentation

IBM produced new user and administration documentation to present the policies, features, and secure uses of the RS/6000 Distributed System. These new documents include the Security Features Users Guide (SFUG) and the Trusted Facilities Manual (TFM).

The SFUG and the TFM are included with the base AIX library, which is installed and available in the evaluated configuration. The AIX library is entirely composed of hypertext mark-up language (HTML) books, as are the SFUG and TFM. This allows the SFUG and the TFM to directly reference the base library information as required. A search engine is also provided as a significant usability enhancement to allow administrators to access documents easily.

8.4.1 Security Features Users Guide

The SFUG consists of 4 chapters. Chapter 1 provides users with an overview of the RS/6000 Distributed System configuration including the hardware and software that comprise the target of evaluation. Chapter 2 describes security evaluations and the impact of evaluations on the assurance users may place in the system to operate according to the stated policy. Chapter 3 describes the security policies enforced by the system (i.e., I&A, DAC, Object Reuse, and Audit) along with a description of the system assurances (i.e., system architecture, system integrity, and system testing). Chapter 4 details how to use the system securely through the use of functions such as logging in and out of the system, changing user identities and passwords, accessing files and devices, handling files and directories, accessing print and network services, and using electronic mail.

This SFUG is written in hypertext mark-up language (HTML) and is viewable from the browser included in the evaluated configuration. The document contains hyperlinks to additional user information including detailed instructions and examples. All these files are shipped with the system and are accessible by any user of the system.

8.4.2 Trusted Facility Manual

The TFM consists of eight sections. The first three sections of the TFM introduce the document, describe security evaluations and the impact of evaluations on the assurance administrators may place in the system, describe the security policies enforced by the and system assurances.

Sections 4 through 7 consist of a series of standard AIX documentation that has been tailored for the evaluated configuration.

- AIX 4.3 Installation Guide; (TFM section 4)
- AIX 4.3 System Management Guide: Operating System and Devices; (TFM section 5)
- AIX 4.3 System Management Guide: Communications and Networks; (TFM section 6)
- AIX 4.3 Guide to Printers and Printing; (TFM section 7)

The tailored versions of these standard documents contain warnings to administrators of what not to do in order to establish and maintain a C2 level of trust system. The TFM also contains the event-specific record format for each audit event.

The TFM is written in hypertext mark-up language (HTML) and is viewable from the browser included in the evaluated configuration. The document contains links to additional user information in the form of detailed instructions and examples through the use of hyperlinks to other RS/6000 Distributed System user documents. All of these administrator documents are shipped with the system and accessible to any user of the system.

8.5 Security Testing

8.5.1 Test Documentation

The vendor test documentation is arranged hierarchically and begins with the RS/6000 Distributed System C2 Evaluation Test Matrix Document (TMD). The Test Matrix Document (TMD) was developed to provide an index into IBM's test documentation and follows the PAT Guidance Working Group Form and Content of Vendor Test Documentation in form and content. The first part of this document describes the vendors test strategy, scope, environment, entry and exit criteria, and lists the C2 level of trust requirements by their individual elements. The second part of this document provides a cross-reference between the C2 level of trust requirements and the TCB interfaces together with their associated test assertions and test strategies and cases.

Beginning with the high-level matrices, C2 level of trust requirements are associated with each TCB interface class (e.g., system calls). In the mid-level matrices, TCB interface classes are broken down into their constituent components and the C2 level of trust requirements are broken down into their individual elements. For example, the TCB interface "system call" is further refined to the system calls (e.g., ***chmod(), creat(), getpriv()***). While the C2 level of trust Requirement "Discretionary Access Control" is further refined to the DAC requirement elements (e.g., DAC-3 The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access.) For each pairing of TCB interface elements and C2 requirement elements in the mid-level matrices there is a reference to a low-level matrix. The low-level matrices provide one or more test assertions for each TCB interface element and C2 requirement element support claim. A test strategy accompanies each test assertion to provide the reviewer with insight into how IBM planned to test the assertion.

Each test case is a grouping of tests that exercise one or more test assertion(s) for the various security relevant functions of the security mechanism interfaces. Test cases that are logically associated and that may be run within a single test build are grouped into test units. A Test Result Journal indicates the associated test assertion, describes the test, and records a PASS/FAIL result for each test case within a test unit. Figure 8-1 illustrates the relationship between these documents.

8.5.2 Test Philosophy

IBM's test philosophy is to test the security mechanisms of the RS/6000 Distributed System by exercising the interface to the TCB and viewing the TCB behavior. Mainly, IBM's approach to testing is black box testing, as defined in section 3.3.1 of *A Guide to Understanding Security Testing and Test Documentation in Trusted Systems*. The testing of some security features, e.g. object reuse, do not lend themselves to this type of testing and so an alternate means is exercised. In most cases, this is performed through an analysis of the system design and/or code.

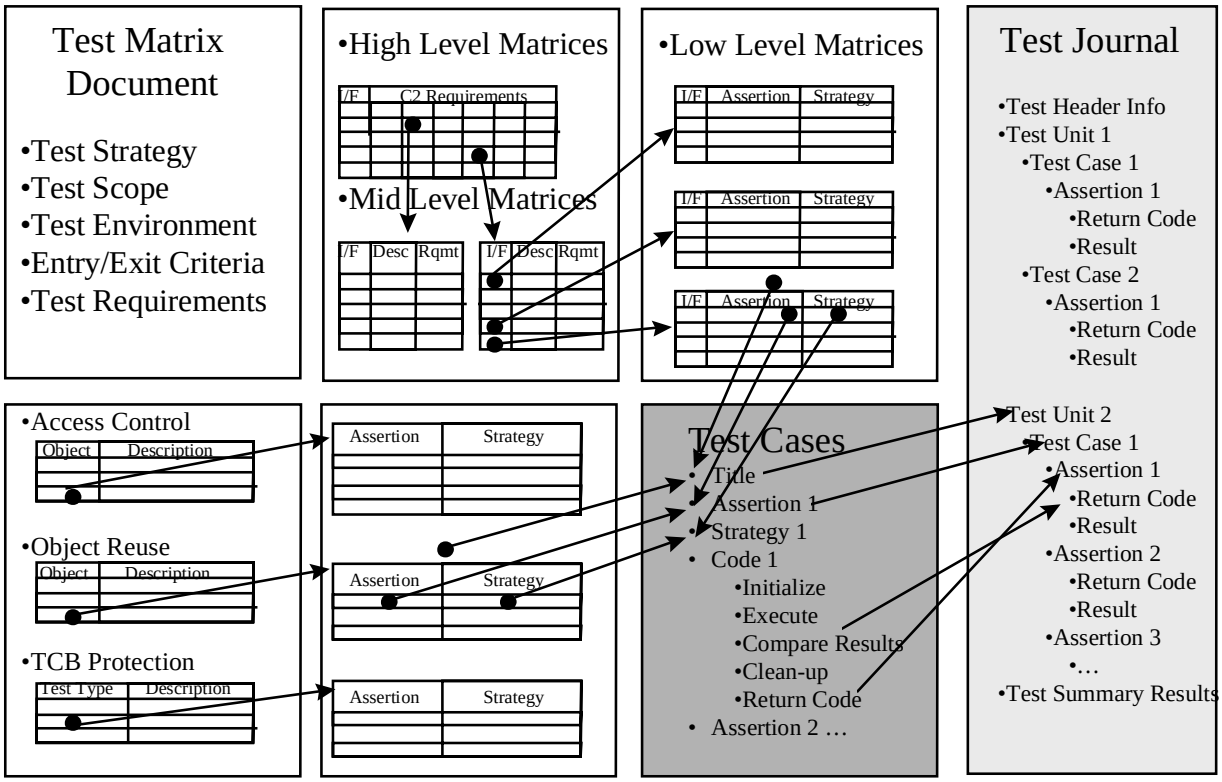

Figure 8-1. Relationships between IBM Test Documentation. *The Test Matrix Document provides an organized view into IBM's traditional test documentation.*

For the testing of object reuse, gray box testing as defined in section 3.3.3 of A Guide to Understanding Security Testing and Test Documentation in Trusted Systems, is used. System integrity is tested through running the diagnostics outside of the secure state, i.e. in single user or "maintenance mode" only.

8.5.3 Test Mechanisms

8.5.3.1 Automation

Most of the tests in the security test suite are automated. Tests that already existed in IBM's test suite and most tests that were added to the RS/6000 Distributed System test suite to satisfy C2 testing requirements are automated. Tests which could not be converted to automatic tests efficiently are manual.

8.5.3.2 Configuration Control

For both automated and manual tests the Test Result Journals for all test cases contain the following information:

- Date and time of the run.
- Tester, driving host (as required), and hostnames of machines used.
- Hardware configuration information from each machine used, i.e. the output of **LSCFG -V** .
- Software configuration information from each machine used, i.e. the output of **LSLPP -L** and **INSTFIX -IK** if required.
- Test journal file - the low level output of the test.
- Test net results - a context sensitive difference between the test journal file and the test baseline file.

8.5.3.3 Test Configurations

The test cases were run on the following configurations. The following is a representative sample of available configurations within the evaluation configuration.

- A stand-alone configuration; administrative master (I&A server) and client are the same machine. (43P, F50, S70)
- A network of 3 machines; one administrative master (I&A server) and 2 clients, and 1 Ethernet LANs and 1 Token Ring; (F50 as administrative master).

8.5.4 Test Coverage by Requirement

The evaluated systems contain no unique hardware or code and are 100 percent commercially available. As such, the quality controls built into the development, test, and service processes for AIX play a role in the functional assurance of the system.

The base release of AIX was subjected to five months of intensive testing by the AIX Product Test Lab (APT). All AIX program temporary fixes generated are regression tested by the APAR Regression Test Lab (ART lab). In addition to APT and the ART lab testing, AIX conforms to standards and has gone through a number of branding tests performed by the Open Systems Conformance Center. The most recent exercise for AIX was UNIX98 branding.

In addition to functional testing, IBM also tested the security mechanisms that support the following C2 policies:

- Discretionary Access Control,
- Identification & Authentication,
- Audit,
- Object Reuse, and
- System Architecture.

Each of these policies is included in the TMD as columns.

The interface that exercises control of these policies is described in the ISD. The evaluation team analyzed the ISD and the system design to ensure that all TCB interfaces were included and accurately defined in the ISD. The following interfaces are included in the TMD as rows.

- System Calls
- User Commands

The individual C2 requirements for each of these policies are reflected in the lower level matrices. Correct enforcement of the C2 policies, as required in the C2 requirements, is described in IBM design documents including the ASD, man pages, and other design documents. Test assertions for each interface are formed from C2 requirements and stated design behavior. Each test assertion is included in one or more test cases. The test cases were run and checked for success.

8.5.4.1 Access Control (DAC)

Discretionary access control tests exist for each interface that controls or specifies access to named objects.

8.5.4.2 Audit

Audit tests exist for each interface that specifies what is audited, cuts audit records, accesses audit data, or performs audit post processing.

8.5.4.3 Identification and Authentication

Identification and authentication tests exist for each interface that provides I&A capabilities or determines access to authentication data.

8.5.4.4 System Architecture

System Architecture tests exist for each interface that enforces the TCB isolation property. The evaluation team reviewed the system architecture for any architectural features that the system depends upon to enforce its security policy and ensured that each of these features was adequately tested. These features include proper access modes on TCB files, and protection of TCB ports (e.g. port 6000).

8.5.4.5 Object Reuse

Object Reuse tests exist for each interface that allocates new resources or that can view the affect of the allocation of new objects. IBM employed the gray-box method of testing by analyzing the methods (initialization, clearing, storage management) employed by the RS/6000 Distributed System for ensuring that residual data was not available to untrusted processes. IBM’s analysis ensured 1) that each of these methods did effectively eliminate residual data and 2) that every allocation of new objects invoked one of these methods.

8.5.5 Evaluation Team Testing

The evaluation team independently executed and reviewed the results of the entire IBM security test suite as well as conducted its own security tests. The evaluation team

- ran vendor test suites on all three hardware platforms and in representative configurations to ensure that the security mechanisms of the product are tested,
- performed ad hoc testing to search for obvious flaws,
- documented and tested assertions to verify vendor test and design documentation, SFUG, TFM, and system integrity tools,
- documented and tested assertions for bringing up the system using the TFM,
- tested the usefulness of the SFUG to an ordinary user, and
- ran representative sample of the system integrity tools.

Details of the team's security testing plan are contained in the evaluation team test report.

9. EVALUATION AS A TCSEC C2 SYSTEM

9.1 Discretionary Access Control

9.1.1 Requirement

The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

9.1.2 Interpretations

I-0002 Delayed revocation of DAC access. A TCB is not required to provide any mechanism for the immediate revocation of DAC access to an object where access has already been established (e.g., opened) when access to that object is reduced. It is sufficient for the SFUG and other documentation to describe the product's revocation policy. However, a change in DAC permissions shall have an immediate effect on attempts to establish new access to that object.

I-0020 DAC authority for assignment. A TCB need not provide all users with the capability to control the sharing of objects. A DAC policy where only system administrators assign access to objects can satisfy the DAC requirement. The SFUG shall clearly identify the roles or user types (e.g., system administrator) who can control sharing.

I-0053 Public objects and DAC. An object for which the TCB unconditionally permits all subjects "read" access shall be considered a public object, provided that only the TCB or privileged subjects may create, delete, or modify the object. No discretionary access checks or auditing are required for "read" accesses to such objects. Attempts to create, delete, or modify such objects shall be considered security-relevant events, and therefore, controlled and auditable. Objects that all subjects can read must be, implicitly, system low.

I-0222 Passwords not acceptable for DAC. The TCB shall not depend solely on passwords as an access control mechanism. If passwords are employed as part of an access control mechanism, they shall not be considered sufficient to satisfy any aspect of the DAC requirement.

I-0226 Initial protection by default (C1-CI-03-86) A system, to pass the Discretionary Access Control requirements at the C2 level and higher, must provide protection by default for all objects at creation time. This may be done through the enforcing of a restrictive default access control on newly created objects or by requiring the user to explicitly specify the desired access

controls on the object when he requests its creation. In either case, there shall be no window of vulnerability through which unauthorized access may be gained to newly created objects.

I-0312 Set-ID mechanism and the DAC requirement. The set-ID mechanism can be part of an acceptable DAC implementation.

9.1.3 Applicable Features

9.1.3.1 Requirement

The named objects in the RS/6000 Distributed System are discussed in Chapter 6. They are ordinary files, directories, block and character device special files, socket special files, FIFOs (named pipes), and System V IPC. A DAC policy is defined for each of these objects. The DAC policy is based on both the user identity and group membership associated with subjects.

All file system objects support both permission bit and Access Control List (ACL) mechanisms as discussed in Chapter 7. The permission bit mechanism allows control of sharing by named individuals, groups, or both, given appropriate administrative setup of groups. The extended permissions mechanism allows this directly, without administrative intervention. In addition, extended permissions permit a list of named individuals and/or groups to be specified with particular modes, including no access.

Propagation of access rights without DAC revalidation is limited to inheritance at process creation. It is not possible to pass access rights (e.g. file descriptors) between unrelated processes.

Named objects are protected from unauthorized access by default through the application of the creator's umask upon creation. The default umask is administratively set to 0700. The user is cautioned in the SFUG to not change the umask. For IPC named objects, the permissions must be set by the creator at creation. If permissions are not specified the access is set to owner only until the owner changes the permissions to allow access.

9.1.3.2 Interpretations

Changes in DAC permissions have an immediate effect on attempts to establish new access to an object. The SFUG describes the revocation policy [I-0002].

For all classes of objects in RS/6000 Distributed System creators have the ability to specify and control sharing of the objects. All users have the capability to control the sharing of any named object they own [I-0020].

Public objects can be read by all subjects and written only by trusted processes. There are no DAC controls on public objects in the RS/6000 Distributed System. Root is the only user that can modify, create or delete public objects [I-0053].

The RS/6000 Distributed System does not rely on passwords as a means of enforcing discretionary access control [I-0222].

FSO objects are protected from unauthorized access by default through the application of the umask and permission bits upon creation. IPC objects are protected from unauthorized access by default through the application of the umask upon creation. Upon creation the creator sets permissions on IPC objects to protect them from unauthorized access. If not set the default is null rendering the object unusable, with the exception of shared memory, which has an interface that allows the creator to modify the permissions. The permissions are stored in the kernel ipc_perm structure [I-0226].

AIX includes a set-ID mechanism [I-0312].

9.1.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Discretionary Access Control requirement.

9.2 Object Reuse

9.2.1 Requirement

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

9.2.2 Interpretations

I-0041 Object reuse applies to all system resources. Analysis for residual data shall be performed on all sharable objects and their attributes (i.e., objects to which MAC or DAC are applied) and other system resources (e.g., stacks, process memory).

9.2.3 Applicable Features

9.2.3.1 Requirement

All resources are protected from Object Reuse (*scavenging*) by one of three techniques: explicit initialization, explicit clearing, or storage management. Initialization sets the resource's contents to a known state before the resource is made accessible to a subject after creation. Clearing sets the resource's contents to a known state when the resource is returned to the TCB for re-use. Storage management ensures that uninitialized storage is never accessible. No information, objects, storage, or buffers in the RS/6000 Distributed System are accessible outside the TCB except to subjects which are authorized such access.

Most resources are explicitly initialized at creation or allocation. Ordinary files, directories, and process memory are initialized whenever a new page is requested, Directory entries are explicitly cleared on deletion. Although not all parts of process context (e.g. floating-point registers) are

saved on every context switch, these are managed to ensure that no subject ever sees "left over" information produced by another subject.

9.2.3.2 Interpretations

Object reuse analysis was performed on all named objects and all storage objects. The RS/6000 Distributed System does not allow any user action prior to I&A other than entering the user name and password [I-0041].

9.2.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Object Reuse requirement.

9.3 Identification and Authentication

9.3.1 Requirement

The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

9.3.2 Interpretations

I-0001 Delayed enforcement of authorization change. If a TCB supports security-relevant authorizations then it shall provide a method for immediate enforcement of removing those authorizations. However, the immediate method (e.g., shutting the system down) need not be the usual method for enforcement. The TFM shall describe both the usual enforcement of granting and removing authorizations and, if the immediate enforcement method is different from the usual one, how an administrator can cause immediate enforcement.

I-0096 Blanking passwords. The TCB shall not produce a visible display of any authentication data entered through the keyboard (e.g., by echoing).

I-0225 Individual accountability in a server environment (C1-CI-02-86) The configuration [an operating system that allows the ability for a user to authenticate himself to the TCB and connect directly to another user's process, if the connecting userid has been specifically granted this access by an authorized user] is allowed. If users of the ADP system are authenticated and have been authorized the ability to connect to another user's process, then DAC is not compromised, provided that the establishment of the path is auditable.

I-0233 Operator logon at system console (C1-CI-04-86) The ‘‘operator’s console” will be considered an exception to the C1, C2, and B1 identification and authentication requirements. The key to this revised definition is the special status of the console, and the physical protection that must be provided. Those consoles outside the physical perimeter of the computer room must be protected by the same physical and procedural mechanisms as the system hardware itself, or must require the use of an identification and authentication mechanism under the control of the TCB.

To meet the various audit requirements, if multiple operators’ consoles are used, the TCB must generate an audit record that identifies from which particular console the auditable event originated. In addition, the trusted facility manual must explain that a log should be maintained of who had access to any particular operator's console at any particular time. Even if only one console is available, such a record is necessary to trace specific events to specific individuals.

I-0234 One-time authentication mechanisms can be acceptable. Single-use authentication mechanisms, such as one-time password devices, can be part of an acceptable identification and authentication mechanism.

I-0240 Passwords may be used for card input. The card input of batch jobs may contain human-readable user passwords. The TFM and the SFUG for the product shall explain the risks in placing passwords on card input and shall suggest procedures to mitigate that risk.

I-0288 Actions allowed before I&A. Prior to having been identified and authenticated by the TCB, a user communicating with the TCB may be allowed to perform only those actions that would not require TCB mediation.

I-0314 Password changes do not require authentication It is not necessary that requests to modify authentication data require reauthentication of the requester's identity at the time of the request.

9.3.3 Applicable Features

9.3.3.1 Requirement

In the RS6000/Distributed System users are required to identify themselves to the system before performing any actions that the TCB is expected to mediate. Authentication data is maintained in the ***/etc/security*** databases. This data includes per-user passwords stored in an encrypted form. The security databases are protected files accessible only to the TCB. User passwords are subject to length and complexity requirements, and can be configured to expire periodically on a per-user basis. User passwords are disabled after too many failed attempts.

For every service request that may result in a change of identity (e.g. telnet**,** FTP, rexec) the TCB performs a login dialogue and validates the user's proposed identity against the authentication provided. Other service requests that do not allow a change in identity (e.g. rsh, rcp, rlogin), the client host passes the LUID to the server, and the server trusts that identity and relies the authentication performed by the client host TCB.

Every user is identified by a unique user name, with which is associated a unique numeric user ID (LUID). This numeric ID is set during the initial login dialogue and is associated with all the subjects created as a consequence of the request (including all the processes created during a login session), and is recorded in the user ID field of audit records.

Technical aspects of these mechanisms are covered in section 5.3.10, Identification and Authentication; section 5.3.11, Interactive Login and Related Mechanisms; and section 7.1, Identification and Authentication (Policy).

9.3.3.2 Interpretations

The RS/6000 Distributed System does not support security-relevant authorizations [I-0001].

The RS/6000 Distributed System does not echo passwords [I-0096].

The RS/6000 Distributed System does not allow a user to authenticate himself to the TCB and connect directly to another user's process [I-0225].

The RS/6000 Distributed System does not have an operator's console [I-0233].

The RS/6000 Distributed System does not include single-use authentication mechanisms [I-0234].

The RS/6000 Distributed System does not support card-based input of batch jobs [I-0240].

The RS/6000 Distributed System does not allow a non-authenticated user to perform any actions that would require TCB mediation [I-0288].

In the RS/6000 Distributed System, requests to modify authentication data require reauthentication of the requester's identity at the time of the request [I-0314]

9.3.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Identification Authentication requirement.

9.4 Audit

9.4.1 Requirement

The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initialization), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be

included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

9.4.2 Interpretations

I-0004 Enforcement of audit settings consistent with protection goals. If the TCB supports the selection of events to be audited, it shall provide a method for immediate enforcement of a change in auditing settings (e.g., to audit a specified user, to audit objects at a particular sensitivity level); however, the immediate method (e.g., shutting the system down) need not be the usual method for enforcement. The TFM shall describe both the usual enforcement of audit settings and, if the immediate enforcement method is different from the usual one, how an administrator can cause immediate enforcement. The TFM shall describe the consequences of changing an audit state dynamically if such changes could result in incomplete or misleading audit data.

I-0005 Action for audit log overflow. When the TCB becomes unable to collect audit data, it shall give a clear indication of this condition and take a pre-specified action. The system implementation may allow an administrator to choose from a range of options. One of the options that may be chosen by the administrator (or, if no choice is available, the only action) shall be that the system cease performing auditable events when the TCB is unable to collect audit data. Choosing an overflow action shall be considered a security-relevant administrative event for the purposes of auditing. The TFM shall fully describe the administrator's options.

I-0006 Audit of user-id for invalid login. When the audit mechanism is required to be capable of producing a record of each login attempt, on failed login attempts it is not required to record in the audit record the character string supplied as the user identity.

I-0043 Auditing use of unnamed pipe. In products that support mechanisms similar to unnamed pipes in UNIX systems, the creation of an unnamed pipe shall be auditable; however, the auditing may be delayed until the pipe becomes sharable with another object (e.g., the creation of a child or the passing of the pipe's descriptor to another subject). At each point that the unnamed pipe is shared, the sharing must be auditable.

I-0073 OK to audit decision regardless of whether action completed. Auditing the attempted introduction of an object at the point of passing the security access control checks satisfies the above requirement, even though the object may not actually be introduced into the subject's address space because of failing later checks not related to security.

I-0208 Auditing of "object not found". (C1-CI-07-84) The ability to audit ''object not found" is not required at the C2 or B1 level. It is required at the B2 level and above if it results in a covert channel.

I-0216 Meaning of "selectively audit''. (C1-CI-02-85) ''Selectively audit" means that the audit mechanism must be sufficiently flexible for the system administrator to obtain information regarding system activity based upon a user's identity or upon object security level. Both of these

capabilities must exist, the ‘‘or” is meant to allow the security administrator the decision to audit one or the other or both. Audit reduction tools, when supplied by the vendor, must be maintained under the same configuration control system as the remainder of the system.

I-0247 Boundaries and documentation for loss of audit data. (C1-CI-02-89) The occurrence of events that may cause audit data to be unpredictably lost must be extremely rare (i.e., not under normal system load) and the number of records lost must be small (i.e., a few bytes as opposed to thousands of bytes). The TFM must completely identify the circumstances of potential audit data loss along with the possible quantity that may be lost. It must be extremely difficult for an untrusted process to cause the loss of audit data. Again, this possibility must be fully defined in the TFM.

The loss of any audit data should be clearly indicated and the action taken should be identified. In addition, the amount of audit data lost must be available. By default, the action taken must be for the system to cease performing events that would have been otherwise audited (e.g., halt temporarily, shut down). The system may allow the administrator to explicitly specify an alternate action to be taken when the condition arises (e.g., continue operating).

I-0286 Auditing unadvertised TCB interfaces. The TCB shall be capable of auditing all security-relevant events that can occur during the operation of the evaluated configuration, including events that may result from the use of TCB interfaces not advertised for general use.

9.4.3 Applicable Features

9.4.3.1 Requirement

Each host in the RS/6000 Distributed System maintains an audit trail file. The trail file is protected by file system permissions. The root user and audit group have read access, and only the root user has write access to the trail file. See section 5.3.15.6, Audit File Protection for further details.

The auditable events include: use of identifications and authentication mechanisms, introduction of objects into a user’s address space, operator and administrator actions, and other events. A summary of the available audit events is located in Appendix G. The origin of I&A requests and object introduction and deletion information is stored as a portion of the variable length record.

The audit record includes the date and time of the event, the user, type of event, and success or failure of the event. The audit record format is located in the Trusted Facilities Manual.

The system administrator can selectively audit on a per-user or per-object basis. Per-user auditing allows the administrator to specify specific classes of audit events that will be recorded for that user. Each process stores a copy of the audit classes that apply to that user as part of the proc table. An audit class is a subset of the total number of audit events available. Per-object auditing allows the administrator to specify certain named objects that will be audited. These objects can be audited based on accesses of a specified mode (read/write/execute) and record the result of the access attempt (success/failure). More information on the use of **AUDIT** commands to accomplish audit selection can be found in section 5.3.15.2, Audit Control and section 5.3.15.4, Audit Record Processing.

9.4.3.2 Interpretations

Immediate enforcement can be forced by rebooting all hosts in the distributed system. The TFM describes the immediate enforcement of audit settings methods**.** If the audit configuration is changed without rebooting all hosts in the network, some user sessions may reflect the old settings and others may reflect the new settings. The TFM describes this potential result and the procedures to correct it [I-0004].

AIX provides BIN mode auditing. If the bin is unable to dump events into the audit trail because it has run out of disk space, it causes a system panic and the system halts. The TFM describes the use of the panic mode administrator option [I-0005].

The RS/6000 Distributed System does not record the character string supplied as the user identity for failed login attempts [I-0006].

In AIX, auditing of unnamed pipes occurs when the pipe is created [I-0043].

Auditing is performed at point of passing the security access control checks. [I-0073].

The RS/6000 Distributed System does not provide the ability to audit ‘‘object not found” [I-0208].

In addition to audit pre-selection, AIX provides tools (specifically, AUDITSELECT) for performing post-selection by user identity [I-0216].

Audit data is only lost if the system fails (i.e., crashes) or runs out of disk space to store audit data. AIX is a robust operating system and not prone to crashes. Thus, it is difficult to cause loss of audit data in this manner. The administrator can place audit data in a file system where non-administrative users do not have the ability to create files, thus reducing the likelihood of running out of disk space. The number of audit records unpredictably lost may be limited by creating small buffers and flushing these often. The worst case loss is 128K per host. The worst case can be mitigated through bin size and byte threshold parameters [I-0247].

There are a small number of TCB interfaces that are not advertised for general use. These have been documented in the ISD, they are tested and the security relevant aspects are auditable [I-0286].

9.4.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Audit requirement.

9.5 System Architecture

9.5.1 Requirement

The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the

TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

9.5.2 Interpretations

I-0213 Administrator interface is part of TCB. Those components of the product that provide the interfaces required for performing administrative actions shall be considered TCB components. The "administrative actions" to which this interpretation applies shall be those that are defined in the TFM to be performed by administrative personnel (e.g., operators, system administrators, system security officers) while the product is in its secure operational state. The TFM shall clearly identify which mechanisms are, and which are not, acceptable for performing each administrative action.

I-0223 Software-Based two-state architectures are acceptable (C1-CI-04-85) Software-based architectures are able to provide process separation and a two state architecture with sufficient assurance to meet the B1 level requirements for System Architecture. Simply because a two-state architecture is provided and maintained primarily by software should not lead to the assumption of its being less secure than hardware in implementing security features.

9.5.3 Applicable Features

9.5.3.1 Requirement

The RS/6000 Distributed System TCB described in section 5.1, TCB Structure, is tamper-resistant because all TCB programs, data, and other components are protected from unauthorized access via numerous mechanisms. The kernel TCB software and data are protected by the hardware memory protection mechanisms. The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes. Non-kernel TCB software and data are protected by DAC, and by process isolation mechanisms. The reserved user ID root owns TCB directories and files. The permissions of the TCB directories and files are set to restrict modification from unauthorized users.

All system-protected resources are managed by the TCB. Because all TCB data structures are protected, these resources can be directly manipulated only by the TCB, through defined TCB interfaces. Resources managed by the kernel software can only be manipulated while running in kernel mode. Processes run in user mode, and execution in the kernel occurs only as the result of an exception or interrupt. The TCB hardware and the kernel software handling these events ensure that the kernel is entered only at pre-determined locations, and within pre-determined parameters. Thus, all kernel-managed resources are protected such that only the appropriate kernel software manipulates them. Trusted processes implement resources managed outside the kernel. The trusted processes and the data defining the resources are protected as described above depending on the type of interface. For directly invoked trusted processes the program invocation mechanism ensures that the trusted process always starts in a protected environment at a

predetermined point. Other trusted process interfaces are started during system initialization and use well-defined protocol or file system mechanisms to receive requests.

9.5.3.2 Interpretations

The administrative interfaces (including WSM) are part of the TCB. The TFM defines those actions which must be performed by the administrator, and which mechanisms are permissible for performing each action [I-0213].

The RS/6000 Distributed System has a hardware-based two-state architecture. [I-0223].

9.5.4 Conclusion

The RS/6000 Distributed System satisfies the C2 System Architecture requirement.

9.6 System Integrity

9.6.1 Requirement

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

9.6.2 Interpretations

I-0144 Availability of diagnostics. If the features provided by the vendor to meet this requirement cannot be exercised by the purchaser of the product, the vendor shall make available appropriate services to use the features as needed to meet the requirement. These services shall be available on an "on-demand" basis.

9.6.3 Applicable Features

9.6.3.1 Requirement

IBM provides software that validates the correct operation of hardware and firmware elements of the TCB. Each of the three models included in the evaluated configuration (43P, F50, and S70) includes Power On Self-Test (POST) firmware in non-volatile memory. The POST runs when each computer is powered-on or reset and performs the following tests:

- registers read and write
- settable bits are set and reset (stuck fault testing)
- level 2 cache read and write (pattern testing)
- memory read and write (pattern testing)
- device controllers read and write(e.g., keyboard, serial port)

Unlike the 43P and F50, the S70 has a separate service processor. The POST test for the S70 performs the following:

- hardware wire tests (e.g., solder joints)
- Built-in Self Test
- service processor test

IBM also provides various diagnostics for the remaining hardware resources of the evaluated configuration. These diagnostics perform the following tests:

- interrupts
- DMA (read and write)
- configuration register read and write
- I/O register read and write
- read/write capability
- input/output capability
- write protect capability
- adapter initialization
- internal loopback (transmit and receive)
- external loopback (transmit and receive)

See section 8.2, System Integrity Tests for more details.

9.6.3.2 Interpretations

The purchaser of the RS/6000 Distributed System can exercise the system integrity features. [I-0144]

9.6.4 Conclusion

The RS/6000 Distributed System satisfies the C2 System Integrity requirement.

9.7 Security Testing

9.7.1 Requirement

The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

9.7.2 Interpretations

I-0170 Functional tests required for object reuse. TCB interface(s) that allow manipulation and review of the contents of a subject's address space and of other resources available at the TCB interface (storage and named objects, device) shall have functional tests included in the vendor test suite to supplement the analysis for object reuse.

9.7.3 Applicable Features

9.7.3.1 Requirement

IBM has submitted a test matrix document, test plan and other test documents containing test requirements, test coverage, test assertions, test procedures, test cases, and test journals for the vendor testing performed on the RS/6000 Distributed System. The evaluation team reviewed these test documents and performed a Test Coverage Analysis. The team worked with IBM to revise the test suite to meet C2 standards.

The evaluation team spent 1 week testing the RS/6000 Distributed System at the IBM facilities in Austin, Texas. The team independently installed the RS/6000 Distributed System Software onto the hardware platforms. The team then executed and reviewed the results of the entire IBM security test suite, as well as conducted its own security tests. All test defects were documented during team testing. IBM revised their test documentation and test suite. A subset of the evaluation team performed a review of the revised documentation to ensure defects were properly addressed and ran a complete regression test on all hardware platforms.

9.7.3.2 Interpretations

The team test report documents the team's testing for object reuse [I-0170].

9.7.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Security Testing requirement.

9.8 Security Features User's Guide

9.8.1 Requirement

A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

9.8.2 Interpretations

I-0244 Flexibility in packaging SFUG. All SFUG documentation shall be in a form that system administrators and users can read at the time that an understanding of the topics covered is needed to use the system in a secure manner (i.e., it shall not be required that the user login in order to read instructions about how to login). The documents or portions of documents that make up the SFUG shall be precisely identified. There are no further restrictions on the packaging (one document, several documents, parts of several documents) or delivery (hardcopy, online) of the SFUG.

9.8.3 Applicable Features

9.8.3.1 Requirement

The RS/6000 Distributed System Security Features User's Guide (SFUG) provides users with an overview of the RS/6000 Distributed System, security evaluations, and the impact of evaluations on the users assurance in the evaluated configuration. It discusses the security policies enforced by the system (i.e., I&A, DAC, Object Reuse, and Audit) along with the system assurances (i.e., system architecture, system integrity, and system testing). The SFUG also contains descriptions of how to use the system securely through various user functions including logging in and out of the system, changing your identity, changing your password, accessing files and devices, handling files and directories, accessing print and network services, and using electronic mail.

9.8.3.2 Interpretations

This SFUG is written in hypertext mark-up language (HTML) and is viewable from the browser included in the evaluated configuration. The document contains hyperlinks to additional user information including detailed instructions and examples. All these files are shipped with the system and are accessible by any user of the system. The TFM directs the administrator to make the SFUG available to users prior to them logging on by printing out a copy of the SFUG and distributing it to each user [I-0244].

9.8.4 Conclusion

The RS/6000 Distributed System satisfies the C2 security features user's guide requirement.

9.9 Trusted Facility Manual

9.9.1 Requirement

A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given.

9.9.2 Interpretations

I-0046 Detailed audit record structure. The documentation of the detailed audit record structure may describe either the raw records produced by the audit mechanism or the output of an audit post-processing tool as long as the records described contain the information specified in the audit requirement. If the output of the audit post-processing tool is described, the tool is considered part of the TCB.

I-0069 Flexibility in packaging TFM. All TFM documentation shall be in a form that system administrators and users can read at the time that an understanding of the topics covered is needed to use the system in a secure manner (i.e., it shall not be required that the user login in order to read instructions about how to bring up the system). The documents or portions of documents that make up the TFM shall be precisely identified. There are no further restrictions on the packaging (one document, several documents, parts of several documents) or delivery (hardcopy, online) of the TFM.

I-0247 Boundaries and documentation for loss of audit data. (C1-CI-02-89) The occurrence of events that may cause audit data to be unpredictably lost must be extremely rare (i.e., not under normal system load) and the number of records lost must be small (i.e., a few bytes as opposed to thousands of bytes). The TFM must completely identify the circumstances of potential audit data loss along with the possible quantity that may be lost. It must be extremely difficult for an untrusted process to cause the loss of audit data. Again, this possibility must be fully defined in the TFM.

The loss of any audit data should be clearly indicated and the action taken should be identified. In addition, the amount of audit data lost must be available. By default, the action taken must be for the system to cease performing events that would have been otherwise audited (e.g., halt temporarily, shut down). The system may allow the administrator to explicitly specify an alternate action to be taken when the condition arises (e.g., continue operating).

9.9.3 Applicable Features

9.9.3.1 Requirement

The RS/6000 Trusted Facilities Manual is intended for system administrators. It provides an overview of the system, describes security evaluations and the impact of evaluations on the administrator's assurance in the system. The TFM describes the evaluated configuration including a listing of the evaluated hardware and software and system documentation. It also discusses the users, subjects, objects and devices analyzed in the evaluation. The TFM also describes the security features provided by the IBM RS/6000 Distributed System running AIX (i.e., the TCB, discretionary access control, object reuse, tamperproof isolation of the TCB and user processes, auditing, and assurances).

The remainder of the TFM provides guidance on the installation, management, administration (network and printing), and auditing of the RS/6000 Distributed System and its TCB hardware and software. Details of these administrative procedures are provided by references to the following documents:

- AIX Version 4.3 Installation Guide
- AIX Version 4.3 System Management Guide: Operating System and Devices
- AIX Version 4.3 System Management Guide: Communications and Networks
- AIX Version 4.3 Guide to Printers and Printing

Details are provided by references to the guides listed above. The guides were written to cover a variety of RS/6000 configurations and uses including hardware and software that are not permitted in the C2 evaluated configuration. The TFM provides a detailed instructions and warnings to administrators for the installation, management, and administration of an evaluated system. The TFM,

- warns the administrator of what not to do (e.g., do not install unevaluated hardware),
- warns the administrator regarding excluded products, services, or hardware (e.g., do not use SMIT for the administration of the system.),
- and provides other relevant information (e.g., possible loss of audit files).

9.9.3.2 Interpretations

The TFM defines the audit record structure as output from the AUDITPR command. AUDITPR is part of the TCB [I-0046].

The “AIX Version 4.3.1 TCSEC Evaluated C2 Security Release Notes” (hardcopy) is included with the RS/6000 Distributed System. This document instructs the administrator how to install the C2 system and gain access to the TFM (HTML). All files that make up the TFM are included in the RS/6000 Distributed System CD [I-0069].

The TFM describes the conditions and circumstances in which audit loss can occur, namely the exhaustion of disk space available to the */audit* directory, system halt, or to a system crash. The TFM explains that in the case of a system crash, all data in physical memory is lost, including any audit records that had not yet been flushed to disk. The audit subsystem enforces a 32K limit on the size of an individual audit record, and only one audit record can be in transit between a thread and the kernel at any given time. The RS/6000 Distributed System TFM includes instructions to the administrator to back up all files, including audit data, on a regular basis to avoid the loss of data due to hard disk failures. The TFM also provides guidance on how to set audit parameters (binsize and bytethreshold) to minimize audit data loss. Based on the recommended setting of these parameters (64K) the amount of audit data that could be lost due to a failure is the sum of the size of those files (128K). [I-0247]

9.9.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Trusted Facility Manual requirement.

9.10 Test Documentation

9.10.1 Requirement

The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

9.10.2 Interpretations

I-0170 Functional tests required for object reuse. TCB interface(s) that allow manipulation and review of the contents of a subject's address space and of other resources available at the TCB interface (storage and named objects, device) shall have functional tests included in the vendor test suite to supplement the analysis for object reuse.

I-0281 Testing system architecture functions. The test plan, procedures, and results shall incorporate tests of the TCB interfaces to mechanisms used to isolate and protect the TCB from external interference.

9.10.3 Applicable Features

9.10.3.1 Requirement

The IBM test documentation is described in section 8.5, Security Testing. This test documentation includes the C2 Evaluation Test Plan, the Test Matrix Document, the Test Cases, and the Test Journals. The test documentation was found to be adequate and sufficient with respect to both breadth and depth of coverage.

9.10.3.2 Interpretations

The team test report documents the team's testing for object reuse [I-0170].

The security testing includes tests of hardware and software mechanisms used to isolate and protect the TCB from external interference. The test plan, test procedures and test results incorporated tests of the TCB interfaces to these mechanisms. Examples include: the protection of privileged ports, the restriction of trusted server connections to trusted clients, properly set protection bits for TCB, memory protection, address space protection, restrictions to only legal instructions and system calls, and protection of privileged instructions from user mode. [I-0281].

9.10.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Test Documentation requirement.

9.11 Design Documentation

9.11.1 Requirement

Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

9.11.2 Interpretations

I-0192 Interface manuals as design documentation. Interface-reference manuals (e.g., UNIX manual pages) are not sufficient, by themselves, as TCB design documentation.

I-0193 Standard system books as design documentation. Books describing the design and implementation of a system used as a basis for a trusted system, but which are inaccurate or incomplete for the trusted system implementation, are not sufficient as design documentation. Such books may partially fulfill the requirement if additional documentation provides a complete description of all differences between the book's description and the actual system implementation, including a satisfactory description of any parts of the TCB not described in the book(s).

9.11.3 Applicable Features

9.11.3.1 Requirement

The RS/6000 Distributed System is a well-documented system. The following documents were written in accordance with the PAT Guidance Working Group Form and Content of Vendor Design Documentation.

- Philosophy of Protection (PoP) for the RS/6000 Distributed System
- Architecture Summary Document (ASD) for the RS/6000.
- Interface Summary Document (ISD) for the RS/6000 Distributed System

Additional design information was available through internal documents. Publicly available documents are listed in Appendix E.

9.11.3.2 Interpretations

Interface-reference manuals constitute a small portion of the design documentation provided for the RS/600 Distributed System [I-0192].

Design and implementation books constitute a small portion of the design documentation provided for the RS/600 Distributed System. Differences (and gaps) between the books description of the implementation and the actual implementation of the RS/6000 Distributed System were resolved (and filled) with additional design documentation. Publicly available examples include programming reference guides, user guides, installation manuals, and interface

specifications. These are listed in Appendix E. Proprietary examples include design specifications, design documents, architecture detailed designs, technical references, the philosophy of protection, the interface summary manual, the architecture summary document, and the TCB identification document. [I-0193].

9.11.4 Conclusion

The RS/6000 Distributed System satisfies the C2 Design Documentation requirement.

APPENDIX A: EVALUATED HARDWARE COMPONENTS

Table A.1 Evaluated Hardware Components.

	43P Model 150	Model F50	Model S70
Machine Type	7043 (PCI Desktop)	7025 (PCI Deskside)	7017 (PCI Rack System)
Product Placement	Workstation	Workgroup Server	Enterprise Server
Physical Configuration	Single PC-style chassis	Single-rack system unit	1 System Rack 1-4 I/O Drawers
System Size	Uniprocessor	Mid-range	High-end
Reference Documentation	7043 43P Models 140 and 240 Service Guide (October 1997)	7025 F50 Service Guide (April 1997)	S70 Installation and Service Guide (October 1997)
Planar	CHRP	CHRP	CHRP
Processor	PowerPC 604e Uniprocessor	PowerPC 604e 4-way SMP	IBM RS64PowerPC up to 12-way SMP Auxiliary Service Processor
CPU Cards	Single CPU on planar • 375 MHz (standard)	2 cards (2 CPUs/card) • 166 MHz F/C 4303 (upgrade to 2-way) and F/C 4309 (second 2-way card). • 332 MHz F/C 4357 (upgrade to 2-way) and F/C 4359 (second 2-way card).	3 cards (4 CPUs/card) • F/C 9404 4-way 125 MHz • F/C 5310 4-way 125 MHz • F/C 5311 4-way 125 MHz
Level 2 Cache	1 MB	256 KB/processor	4 MB/processor
Memory Configuration	128 MB minimum installed, expandable to 1 GB. 1 to 4 168-pin DIMMs with ECC: • F/C 4149 64 MB • F/C 4150 128 MB • F/C 4169 256 MB	128 MB minimum installed, expandable to 3 GB. 2-32 200-pin DIMMs with ECC: • F/C 9083 Base 128 MB • F/C 4107 64 MB • F/C 4110 256 MB • F/C 4106 256 MB	R1 Memory Groups, 512 MB minimum installed, expandable to 16 GB, as follows: • F/C 9168 Base 512MB plus 0-4 F/C 4171 512MB • F/C 4174 1024MB plus 0-4 F/C 4173 1024MB • F/C 4176 2048MB plus 0-4 F/C 4175 2048MB • F/C 4178 4096MB plus 0-3 F/C 4177 4096MB
Bus Architectures	• PowerPC local bus (64 bit) • 1 PCI bus (32 bits 4 slots)	• PowerPC local bus (128 bit) • 3 PCI buses (2 64-bit, 7 32-bit slots total) • ISA bus (2 slots)	• PowerPC local bus (dual 512 bit data, address buses) • 4 PCI buses per I/O Drawer (5 64-bit, 9 32-bit slots/rack) • 4 RIO buses between system and I/O Drawers

Table A.1 cont. Evaluated Hardware Components.

Table A.1 (cont.)	43P Model 150	Model F50	Model S70
SCSI Configuration (at least one required for each computer)	• 1 PCI Ultra-SCSI on planar • 0-2 F/C 6206 PCI Ultra-SCSI adapters	• 2 PCI SCSI-2 (Fast/Wide) on planar • 0-6 F/C 6206 PCI Ultra-SCSI adapters	• 1-8 F/C 6206 PCI Ultra-SCSI adapters per I/O Drawer
SCSI Storage Devices (0+ means zero or more; 1+ means one or more)	4 SCSI bays available: • F/C 2900 or 2908 DASD (1+) • F/C 2624 CDROM (1+) • F/C 6159 4mm Tape (0+)	21 SCSI bays available: • F/C 3080, 3084, 2900, 2911, or 3019 DASD (1+) • F/C 2619 CDROM (1+) • F/C 6159 4mm Tape (0+)	up to 16 total SCSI drives/drawer: • F/C 9394, 2900, 2911, or 3019 DASD (1-12) • F/C 2619 CDROM drives (1+) • F/C 6159 4mm tape drives (0+)
Diskette Drive	1.44 MB diskette	1.44 MB diskette	1.44 MB diskette
Graphics Adapters, PCI 2-D (one required)	• F/C 2838 GXT120P	• F/C 2838 GXT120P	• F/C 2838 GXT120P
Monitor (one required)	• F/C 3626 P202 (21") • F/C 3620 P72 (17")	• F/C 3626 P202 (21") • F/C 3670 P72 (17")	• F/C 3626 P202 (21") • F/C 3620 P72 (17")
Network adapters (at least one required for each computer)	• Ethernet onboard equivalent to F/C 2968 • F/C 2968 Ethernet • F/C 2975 Token Ring • F/C 2987 Ethernet	• Ethernet onboard equivalent to F/C 2987 • F/C 2968 Ethernet • F/C 2979 Token Ring • F/C 2987 Ethernet	• F/C 2968 Ethernet • F/C 2979 Token Ring • F/C 2987 Ethernet
Keyboard, Mouse (one each required)	• F/C 6600 Keyboard • F/C 6041 Mouse	• F/C 6600 Keyboard • F/C 6041 Mouse	• F/C 6600 Keyboard • F/C 6041 Mouse
Operator Panel, Controls	• Operator Panel Display (LED) • Power Switch w/LED • Reset Button	• Operator Panel Display (LED) • Power Switch w/ LED • Reset Button	• Operator Panel Display (LED) • Scroll/Enter Pushbuttons • Attention/Power LEDs • Power Push-button • Service Processor (requires external monitor).
Service Processor	No	Yes	Yes
Printer (optional)	IBM Model 4317 (Network Printer 17)	IBM Model 4317 (Network Printer 17)	IBM Model 4317 (Network Printer 17)

APPENDIX B: EVALUATED SOFTWARE

The evaluated software product is AIX Version 4.3.1 TCSEC Evaluated C2 Security. To meet the C2 requirements, the administrator shall follow the *AIX Version 4.3.1 TCSEC Evaluated C2 Security Release Notes* detailing the initial system setup and then follow the additional TFM guidance on installing and maintaining a C2 system.

APPENDIX C: ACRONYMS

ACL	Access Control List
AIX	Advanced Interactive eXecutive
APAR	Authorized Program Analysis Report.
API	Application Programming Interface
ASD	Architecture Summary Document
CMVC	Configuration Management Version Control
DAC	Discretionary Access Control
DASD	Direct Access Storage Device
DIMM	Dual Inline Memory Module
EPL	Evaluated Product List
EUID	Effective User ID
FTP	File Transfer Protocol
GID	Group Identifier
GUI	Graphical User Interface
I&A	Identification and Authorization
IP	Internet Protocol
IPC	Inter-Process Communications
ISD	Interface Summary Document
ITSEC	Information Technology Security Evaluation Criteria
JFS	Journaled File System
LAN	Local Area Network
LFS	Logical File System
LPP	Licensed Program Product.
NFS	Network File System
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PGWG	Process Action Team (PAT) Guidance Working Group
POST	Power On Self Test
PoP	Philosophy of Protection
PTF	Program Temporary Fix
RFC	Request for Comments
rlogin	Remote login
RPC	Remote Procedure Call
SFUG	Security Features User's Guide

SMP	Symmetric Multi-Processing
SMIT	System Management Interface Tool
TAI	Trusted Application Interface
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TCSEC	Trusted Computer System Evaluation Criteria
ToE	Target of Evaluation
TEF	TTAP Evaluation Facility
TFM	Trusted Facility Manual
TPEP	Trusted Products Evaluation Program
TRB	Technical Review Board
TTAP	Trusted Technology Assessment Program
TU	Test Unit
UDP	User Datagram Protocol
UID	User Identifier
UP	Uni-Processor
WSM	Web-based System Management

APPENDIX D: GLOSSARY

AIX	Advance Interactive eXecutive; IBM's version of UNIX for the RS/6000 computer.
APAR	Authorized Program Analysis Report. A description of a problem with IBM software that includes symptoms, levels of the software to which it applies, severity, work-arounds if possible, and a description of the solution. When IBM supplies a software fix for the problem the APAR will also list the current PTFs that are to be applied to fix the problem.
Client	A participant that makes a request of a server role entity using a client-server protocol.
Client-server model	An architecture used in network applications to permit client and server programs to run on different computers. The client program handles user requests and makes service requests to the server program.
CMVC	Configuration Management Version Control. An IBM LPP that does software lifecycle project management. It tracks defects, source code, builds, and testing.
DAC	Discretionary Access Control. A means of restricting access to objects based on the identity and need-to-know of the user, process and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.
Daemon	A process, not associated with a particular user, that provides services to the local operating system or its users.
EPL	Evaluated Product List. A list of equipment, hardware, software, and/or firmware that has been evaluated against, and found to be technically compliant, at a particular level of trust, with the DoD TCSEC by the NCSC. The EPL is included in the National Security Agency Information Systems Security Products and Services Catalogue.
FTP	File Transfer Protocol. The predominant internet protocol used to transfer files between hosts.
Host	A term used to refer to one of the multiple RS/6000 computers that makes up the RS/6000 system.
I&A	The identification by which an individual requests a session on a system (logs in), and the verification that the individual is who he/she claims to be.
IP	Internet Protocol. The network layer protocol that is used to route data from an origin host to a destination host on the Internet or on an intranetwork.
IPC	Inter-Process Communications. Specifically, the System V IPC mechanisms that includes message queues, semaphores, and shared memory segments.
LPP	Licensed Program Product. A software product that is separately order-able

	with unique terms and conditions or is aggregated with other programs as a "feature" of the aggregate.
Local Host	(1) From the perspective of a user, the host computer to which the user logged in; (2) From the perspective of an executing program, the host on which the program is running; (3) 'localhost,' a hostname alias for the local host computer.
Named Object	An object that can be accessed by multiple user processes using a naming convention provided at the TCB interface.
NFS	Network File System. NFS is the protocol used between hosts to permit a user at one host to access files physically located on another host.
Object Reuse	The reassignment and reuse of a storage medium (e.g., page frame, disk sector, magnetic tape) that once contained one or more objects. To be securely reused and assigned to a new subject, storage media must contain no residual data (magnetic remnant) from object(s) previously contained in the media.
PGWG	Process Action Team (PAT) Guidance Working Group; NSA-sponsored group that wrote the "Form and Content" requirements for vendor design and test documentation.
PoP	Philosophy of Protection. An informal description of the overall design of a system that delineates each of the protection mechanisms employed. A combination (appropriate to the evaluation class) of formal and informal techniques is used to show that the mechanisms are adequate to enforce the security policy.
PTF	Program Temporary Fix. A software fix for a problem described by an APAR built at and for a specific version of the software. Because multiple versions of a software product may exist concurrently many PTFs may be generated by a single problem. Also depending on the size and resultant packaging of the fix, multiple PTFs may be generated for each version of software.
Public Object	An object that can be publicly read by any user in the system, but can be modified only by administrative users.
RFC	Request for Comments. A form of protocol documentation.
rlogin	Remote login. A service originally implemented on Berkeley UNIX systems that allows authorized users of one host to connect to another host across a network connection and interact as if their terminals were connected directly.
RPC	Remote Procedure call. (1) In general, a facility that permits a local program to request that a procedure call be executed on a remote host. (2) The means used to request various Berkeley UNIX services, such as NFS, and mountd.
Service Protocol	The networking protocol used by a client application to request services from a server. This is a high level protocol that maps to the Presentation and Application layers of the OSI reference model.

SFUG	Security Features User's Guide. A document necessary to satisfy the requirements of any TCSEC class. The SFUG is directed toward the general users of the system. It describes the protection mechanisms provided by the TCB, contains guidance on their use, and describes how they interact with one another.
SMP	Symmetric Multi-Processing. A tightly-coupled multi-processor computer system that functions as if it were a singly computer (single copy of code, shared data structures). The ToE includes up to 12-way SMP.
Socket	(1) An abstraction used by Berkeley UNIX that allows an application to access TCP/IP protocol functions. (2) An IP address and port number pairing.
Storage Object	An object that supports both read and write accesses. (source: TCSEC glossary).
Subject	An active entity, generally in the form of a person, process, or device that causes information to flow among objects or changes the system state. Technically, a process/domain pair. (source: TCSEC glossary).
System	For the RS/6000 evaluation, the system refers to any composition of one to all of the RS/6000 host systems, and the associated LAN hardware used to connect them, that make up the Target of Evaluation. This assembly of hardware and software, when connected and administered as defined in the [RS-TFM], enforces a well-defined security policy for the resources protected by the system.
TAI	Trusted Application Interface. A privileged interface that is intended for use with an application that is trusted with the capability to bypass the security mechanisms of the product.
TCB	Trusted Computing Base. The totality of protection mechanisms within a computer system - including hardware, firmware, and software - the combination of which is responsible for enforcing a security policy. A TCB consists of one or more components that together enforce a unified security policy over a product or system. The ability of a TCB to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters (e.g. a user's clearance) related to the security policy [TCSEC].
TCP	Transmission Control Protocol. A connection-oriented transport layer protocol that provides sequencing, error recover, and flow control.
TCSEC	Trusted Computer System Evaluation Criteria. A document published by the National Computer Security Center containing a uniform set of basic requirements and evaluation classes for assessing degrees of assurance in the effectiveness of hardware and software security controls built into systems. These criteria are intended for use in the design and evaluation of systems that will process and/or store sensitive or classified data. This document is Government Standard DoD 5200.28-STD and is frequently referred to as "The Criteria" or "The Orange Book."

TEF	Licensed TTAP Evaluation Facility, for example, Arca.
Telnet	An application protocol that provided interactive access to a remote host.
TFM	Trusted Facility Manual. A document necessary to satisfy the requirements of any TCSEC class. The TFM is directed towards administrators of an installation. It provides detailed information on how to: (1) configure and install the secure system; (2) operate the system securely; (3) correctly and effectively use system privileges and protection mechanisms to control access to administrative functions; and (4) avoid improper use of those function which could compromise TCB and user security.
ToE	Target of Evaluation. The specific configuration of hardware, firmware, and software that is the candidate for an evaluation rating. Sometimes described as "evaluation configuration."
TPEP	Trusted Products Evaluation Program. The NSA program to evaluate trusted products developed to meet a level of assurance specifies in the TCSEC. Evaluated products are place on the evaluated products list (EPL) which is part of the National Security Agency Information System Security Products and Services Catalogue.
TRB	Technical Review Board. The panel of senior evaluators which the evaluators present evidence during the NCSC evaluation process. A TRB meets at the end of each of the two stages of evaluations: Design Review and Formal Evaluation. The TRB generates a recommendation at each stage. These recommendations fall into three categories: (1) pass, (2) pass with some exceptions, and (3) fail. The actual outcome is decided by NCSC management, who use the TRB results as input into their decision process.
TTAP	Trusted Technology Assessment Program. The National Security Agency implemented TTAP in January of 1997 to authorize and oversee commercial facilities performing trusted product evaluations.
TU	Test Unit. A uniquely dispatch-able unit of testing work. It may include many test cases and test variations but generally has a common thread, e.g. a login auditing TU.
UDP	User Datagram Protocol. A connectionless transport layer protocol, built directly on the IP layer, that sends a single datagram to a remote host. It does not provide sequencing, error recovery, nor flow control.
UP	Uni-Processor. As contrasted with SMP, a host computer with a single processor.
User	Any authorized or unauthorized person who interacts directly with a system; not including either passive or active wiretappers on the LAN or persons who physically attack the computer system.
WSM	(Web-based Systems Management.) A utility used by administrators for setting up and configuring AIX systems.
Workstation	One of the multiple RS/6000 computers that makes up the RS/6000 system.

APPENDIX E: REFERENCES

Advanced Micro Devices, *The Open Programmable Interrupt Controller (PIC) Register Interface Specification Revision 1.2,* AMD, 1 October 95.

DeRoest, James. *AIX Version 4 System and Administration Guide* J. Ranade Workstation Series McGraw-Hill New York, New York. 1997.

Faigan, D. P. et. al. *A Rigorous Approach to Determining Objects* Proceedings of the 8th Annual Computer Security Applications Conference. December 1993.

IBM, *1080 MB and 2160 MB SCSI-2 Disk Drives Installation User's Guide,* IBM Corp., 1 April 97.

IBM, *10/100 Mbps Ethernet-TX PCI Adapter Installation and User's Guide,* IBM Corp., 1 October 97.

IBM, *20X (Max) SCSI-2 CD-ROM Drive Installation and User's Guide,* IBM Corp., 1 October 97.

IBM, *4mm Tape Drive Installation Guide,* IBM Corp., 1 November 96.

IBM, *AIXv3 Operating System Technical Reference,* Volume 3 Chapters 1-12, Advanced Engineering Systems, IBM Corp., 9 March 92.

IBM, *AIX General Programming Concepts: Writing and Debugging Programs, Version 4* 4th ed. IBM Corp., April 97.

IBM, A*IX Kernel Extension and Device Support Programming Concepts, Version 4* 5th ed. IBM Corp., April 97.

IBM, A*IX System Management Guide: Operating System and Devices, Version 4* 5th ed. IBM Corp., April 97.

IBM, *AIX Technical Reference: Master Index, Version 4* 3d ed. IBM Corp., October 96.

IBM, *AIX Technical Reference: Volume 8: Enhanced X-Windows, Version 4* 3d ed. IBM Corp., April 96.

IBM, *AIX Technical Reference: Volume 9: Enhanced X-Windows, Version 4* 2d ed. IBM Corp., April 96.

IBM, *AIX Technical Reference: Volume 10: Enhanced X-Windows, Version 4* 2d ed. IBM Corp., April 96.

IBM, *AIXv4 Device Configuration Developers Guide,* Draft 2.1 IBM Corp., 16 January 97.

IBM. *AIX v4 Device Configuration Developers Guide,* Draft 2.1 Advanced Workstation Systems, IBM Corp., 16 January 97.

IBM, *Ethernet PCI Adapter Installation Guide*, IBM Corp., 1 November 96.

IBM, *PCI Differential Ultra SCSI Adapter Installation and User's Guide*, IBM Corp., 1 April 97.

IBM, *PCI SCSI-2 Fast/Wide Differential Adapter Installation Guide*, IBM Corp., 1 November 96.

IBM, *PCI SCSI-2 Fast/Wide Single-Ended Adapter Installation Guide*, IBM Corp., 1 November 96.

IBM, *PCI Single-Ended Ultra SCSI Adapter Installation and User's Guide*, IBM Corp., 1 April 97.

IBM, *PCI Adapter Placement Reference*, IBM Corp., 1 October 97.

IBM, *Power GXT120P Graphics PCI Adapter Installation and User's Guide*, IBM Corp., 1 October 97.

IBM, *PowerPC Microprocessor Common Hardware Reference Platform (CHRP) System binding to: IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware Revision 1.8*, IBM Corp., 2 February 98.

IBM, *RS/6000 7043 43P Series Models 140 and 240 Service Guide*, IBM Corp., 1 October 97.

IBM, *RS/6000 7043 43P Series Models 140 and 240 User's Guide*, IBM Corp., 1 October 97.

IBM, *RS/6000 7025 F50 Series Service Guide*, IBM Corp., 1 April 97.

IBM, *RS/6000 7025 F50 User's Guide*, IBM Corp., 1 April 97.

IBM, *RS/6000 Adapters, Devices, and Cable Information for Multiple Bus Systems*, IBM Corp., 1 October 97.

IBM, *RS/6000 Enterprise Server S70 Installation and Service Guide*, IBM Corp., 1 October 97.

IBM, *RS/6000 Enterprise Server S70 Technology and Architecture*, IBM Corp., 1 April 98.

IBM, *RS/6000 Enterprise Server S70 User's Guide*, IBM Corp., 1 October 97.

IBM, *RS/6000 Workstations: Facts and Features*, IBM Corp., February 97.

IBM, *SCSI-2 Fast/Wide Disk Drives ULTRA SCSI Fast/Wide Disk Drives Installation and User's Guide*, IBM Corp., 1 April 97.

IBM, *SCSI-2 Fast/Wide RAID Adapter Installation Guide*, IBM Corp., 1 June 96.

IBM, *SCSI-2 Fast/Wide Hot-Swap Disk Drive*, IBM Corp., 1 November 96.

IBM, *Service Processor 1.1 Installation and User's Guide*, IBM Corp., 1 October 1997.

IBM, *Token-Ring PCI Adapter Installation and User's Guide*, IBM Corp., 1 October 1997.

IBM, *Trusted Facility Manual (TFM) for the RS/6000 Distributed System* IBM Corp., 1998.

IBM, *Security Features User's Guide (SFUG) for the RS/6000 Distributed System* IBM Corp., 1998.

IEEE 1275 Working Group, *PCI Bus Binding to: IEEE Std 1275-1994 Standard for Boot (Initialization Configuration) Firmware Revision 2*, IEEE. 7 August 96.

Kelly, David A. *AIX/6000: Internals and Architecture*. J. Ranade Workstation Series. McGraw-Hill New York, New York 1996.

May, Cathy. et. al. *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, 2d ed. Morgan Kaufmann Publishers, Inc. San Francisco, California.1994.

Motorola, *PowerPC 604 RISC Microprocessor User's Manual*, Motorola. 1 November 1994.

Motorola, *Addendum to PowerPC 604 RISC Microprocessor User's Manual: PowerPC 604e Microprocessor Supplement and User's Manual Errata,* Motorola, 1996.

National Computer Security Center, PAT Guidance Working Group. *Form and Content of Vendor Design Documentation,* Washington, May 1994.

Nye, Adrian, *X Protocol Reference Manual*. O'Reilly & Associates, Inc. Sebastopol, California. 1995.

O'Quin, John T. *AIX CHRP Architecture, Version 0.6,* IBM Corp., July 11, 1996

PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture. Morgan Kaufmann Publishers, Inc., San Francisco, CA 1995.

PowerPC Microprocessor Family: The Programming Reference Guide, Apple/IBM/Motorola 1995.

PowerPC Microprocessor Family: The Programming Environments, Apple/IBM/Motorola 1997.

Sieger, Adreas. *The AIX Survival Guide.* Addison-Wesley, November 1996.

Stern, Hal. *Managing NFS and NIS*. O'Reilly & Associates, Inc. Sebastopol, California. 1991

Stevens, Richard W. *UNIX Network Programming.* Prentice Hall Software Series Prentice-Hall, Inc. Englewood Cliffs, New Jersey. 1990.

U.S. Department of Defense, *Trusted Computer System Evaluation Criteria*, DOD5200.28-STD. Washington, December 1985.

APPENDIX F: EPL Entry

Report No. CSC-EPL-98-004

AS OF: 18 December 1998

PRODUCT: RS/6000 Distributed System

VENDOR: IBM Corporation

CANDIDATE CLASS: C2

TTAP EVALUATION FACILITY:

Arca Systems Evaluation Facilities (San Jose, Boston, Washington D.C., Austin), www.arca.com. Contact Doug Landoll at (512) 310-2228, landoll@arca.com.

PRODUCT DESCRIPTION:

The RS/6000 Distributed System is a collection of IBM RS/6000 host computers connected via a physically protected Local Area Network (LAN). The RS/6000 is a line of high-performance Uni-Processor (UP) and Symmetric Multi-Processing (SMP) computers based on 32-bit and 64-bit PowerPC processors and intended for use as a closed distributed network of workstations and servers.

The target of evaluation consists of multiple interconnected RS/6000 hosts, each running Release 4.3.1 Evaluated C2 Security. AIX is IBM's version of UNIX, which is differentiated from other UNIX products by its system administration tools, Journaled File System (JFS), pageable/preemptable kernel, loadable kernel extensions, hardware error detection, and available applications. AIX Version 4.3 is POSIX- and SVID-compliant. All hosts include a X Windows System with 2-D graphics capability, for local use. The X Windows system is considered trusted only when invoked by the administrator to perform administrative actions through wsm.

Each host provides the same set of local services (such as file, memory, and process management) and also provides network services (e.g., remote logins, file transfers, and network file services) to users on other hosts within the distributed system.

PRODUCT STATUS:

AIX has been sold and supported as a commercial-off-the-shelf product by IBM Corp., since 1986. As of 1st quarter 1999, IBM plans to announce the availability of the RS/6000 Distributed System and AIX Release 4.3.1 Evaluated C2 Security.

SECURITY EVALUATION STATUS:

The security protection provided by the RS/6000 Distributed System when configured according to the Trusted Facility Manual has been evaluated by Arca Systems. The security features of the RS/6000 Distributed System have been examined against the requirements specified by the Department of Defense TCSEC dated December 1985.

The Arca Systems evaluation team has determined the RS/6000 Distributed System satisfies all the specified requirements of the criteria at class C2. For a complete description of how the RS/6000 Distributed System satisfies each requirement of the Criteria, see Final Evaluation Report, IBM Corporation, RS/6000 Distributed System running AIX 4.3.1 Evaluated C2 Security (Report CSC-FER-98/004).

A system that has been rated C2 enforces a discretionary access control policy to protect information. It allows users to share information under their control only with other specified users. It helps identify and authenticate users in order to control access to the system and enforces accountability. It prevents access to residual information from a previous user's actions, and provides for the auditing of security related events. The hosting hardware for the RS/6000 Distributed System in the evaluated configuration is the IBM 43P, F50, and S70.

ENVIRONMENTAL STRENGTHS:

The RS/6000 Distributed System provides user identification and authentication (I&A) by requiring each user to login with the proper password at the local workstation and also at any remote host, where the user can enter commands (e.g., remote login, telnet sessions). The system maintains a consistent administrative database by making all administrative changes on a designated Network File System (NFS) server and exporting the database files to all hosts in the system. Thus, a user ID on any host refers to the same individual on all other hosts. Each host enforces a coherent discretionary access control (DAC) policy based on UNIX-style mode bits, and an optional Access Control List (ACL), for those named objects under its control. Each host performs its own auditing using an audit configuration that is consistent across the system.

VENDOR CONTACT:

The RS/6000 Distributed System will be available through Product Request for Price and Quotes. The point of contact for the evaluation is Margie Belvins, (512) 838-3176.

APPENDIX G. AUDIT RECORD FORMAT

This appendix defines the format of an audit trail and each audit record within the trail. The audit record format consists of a fixed-format header and a type-specific tail.

Audit Trail Format

An audit trail consists of a sequence of bins. Each bin starts with a bin head and must be terminated by a bin tail before other bins can be appended to the trail.

Table G.1: Audit Trail Bins

Name	Description
bin_magic	The magic number for the bin, 0xf0f0.
bin_version	The version number for the bin (3).
bin_tail	Indicates whether the bin describes the audit trail head or tail: 0 = bin header 1 = bin end (tail) 2 = trail end
bin_len	The (unpacked) length of the bin's records. A nonzero value indicates that the bin has a tail record.
bin_plen	The current length of the bin's record (might be packed).
bin_time	The time at which the head or tail was written.
bin_cpuid[8]	CPU id

Audit Record Header Format

The following table contains the header format for each audit record in an audit trail. This is defined in /usr/include/audit.h.

Table G.2: Audit Header Format

Name	Description
ah_magic	Magic number for audit record.
ah_length	The length of the tail portion of the audit record.
ah_event[16]	The name of the event and a null terminator.
ah_result	An indication of whether the event describes a successful operation. The values for this field are: 0 = successful completion 1 = failure >1 = an errno value describing the failure
ah_ruid	The real user ID; that is, the ID number of the user who created the process that wrote this record.
ah_luid	The login ID of the user who created the process that wrote this record.
ah_name[16]	The program name of the process, along with a null terminator.
ah_pid	The process ID of the process that wrote this record.
ah_ppid	The process ID of the parent of this process.
ah_tid	The thread ID.
ah_time	The time in seconds at which this audit record was written.
ah_ntime	The nanoseconds offset from ***ah_time***.
ah_cpuid[8]	CPU identifier.

Audit Tail Format

This section contains the event-specific record format for each audit event. The audit tail varies from one event to another, but each event uses the following set of format definitions to describe the layout of audit tail

Table G.3: Audit Tail Format

Format	Size	Description
%A	Variable length byte stream.	First 32-bit word N gives total length in bytes with N additional bytes (for a total size of `N + 4` bytes of data in the event record) representing the access control list. Formatted output is as with ***aclget*** command.
%d	1 32-bit word.	Formatted as 32-bit signed decimal integer.
%G	Variable length byte stream.	Formatted as comma-separated list of group names.
%L	Socket description with login ID.	Internet Domain Sockets: Formats as a string with the requester given as ***user@host.domain*** and the requester's port number as a decimal value. Unix Domain Sockets: LUID is ignored, and the pathname associated with the socket is formatted as a string.
%o	1 32-bit word.	Formatted as 32-bit octal integer.
%P	Variable length byte stream.	First 32-bit word N gives total length in bytes with N additional bytes (for a total size of `N + 4` bytes of data in the event record) representing the privilege control list. Formatted output is as with ***pclget*** command.
%s	Variable length string. May contain one or more NUL characters.	Formatted as a text string.
%S	Socket description.	Internet Domain Sockets: Formats as an `IP` address and port number or service name. Unix Domain Sockets: The pathname associated with the socket is formatted as a string.
%T	2 32-bit words. First 32-bit word represents whole seconds since the epoch. Second 32-bit words represents microseconds.	Formatted as text string giving date and time with 6 significant digits for the seconds (`DD Mmm YYYY HH:MM:SS.mmmuuu`).
%x	1 32-bit word.	Formatted as 32-bit hexadecimal integer.

Table G.4: Process Events

Audit Event	System Call	Description	Detailed Record Info
PROC_Create	***fork()***	A new process is created.	forked child process %d
PROC_Delete	***exit()***	A process is terminated.	exited child process %d
PROC_Execute	***exec()***	A new program is executed.	effective user id %d effective group id %d privilege %x:%x filename %s
PROC_RealUID	***setuidx()***	One or more user id values for a process are changed.	old real user id %d
PROC_AuditID	***setuidx()***	One or more user id values for a process are changed.	old login user id %d
PROC_SetUserIDs	***setuidx()***	One or more user id values for a process are changed.	effective user id %d real user id %d saved user id %d login user id %d
PROC_RealGID	***setgidx()***	One or more group id values for a process are changed.	real group id %d
PROC_Environ	***usrinfo()***	Various pieces of user environment data were changed.	environment %s
PROC_Limits	***setrlimit()***	The resource limits for a process were set.	
PROC_SetPri	***nice()***	The process niceness for a non-fixed priority process was set.	nice value %d
PROC_Setpri	***setpri()***	A fixed priority for a process was set.	fixed priority %d
PROC_Privilege	***setpriv()***	One or more privilege vectors for a process were changed.	command %x privilege set %x:%x
PROC_Settimer	***settimer()***	One of the system timers were changed.	old time %T new time %T
PROC_Adjtime	***adjtime()***	The system clock was changed.	old time %T delta %x:%x
PROC_Debug	***ptrace()***	Debugging operations were performed on a process.	process id %d command %d
PROC_Kill	***kill()***	Send a signal to a process.	process id %d signal %d
PROC_Setpgid	***setpgid()***	The process group ID was set.	process id %d process group %d
PROC_Load	***ld_loadmodule()***	A new object module was loaded into the process address space.	file %s
PROC_LoadMember	***ld_loadmodule()***	A new object module was loaded into the process address space.	file %s member %s
PROC_LoadError	***ld_loadmodule()***	A new object module was loaded into the process address space.	flags %x library path %s file %s
PROC_SetGroups	***setgroups()***	The process concurrent group set was changed.	group set %G
ACCT_Disable	***acct()***	System accounting was disabled.	
ACCT_Enable	***acct()***	System accounting was enabled.	file %s

Table G.5: File System Events

Audit Event	System Call	Description	Detailed Record Info
FILE_Open	***open()*** ***creat()***	A new file descriptor was created for a named file.	flags %o mode %o filename %s
TCB_Leak	***open()*** ***creat()***	A file with the TCB attribute has been opened for read.	
TCB_Mod	***open()*** ***creat()***	A file with the TCB attribute has been opened for write.	
FILE_Read	***read()***	A file descriptor was read from.	file descriptor %d
FILE_Write	***write()***	A file descriptor was written to.	file descriptor %d
FILE_Close	***close()***	A file descriptor was closed.	file descriptor %d
FILE_Link	***link()***	A new directory entry was created for a file.	linkname %s filename %s
FILE_Unlink	***unlink()***	A file system object was removed.	filename %s
FILE_Rename	***rename()***	A file system object's name was changed.	old name with path %s new name with path %s
FILE_Owner	***chown()***	A file's ownership was changed.	owner %d group %d filename %s
FILE_Mode	***chmod()***	A file's mode was changed.	mode %o filename %s
FILE_Fchmod	***fchmod()***	An open file's mode was changed.	mode %o file descriptor %d
FILE_Fchown	***fchown()***	An open file's ownership was changed.	owner %d group %d file descriptor %d
FILE_Truncate	***truncate()***	Portions of a file were removed.	filename %s
FILE_Symlink	***symlink()***	A symbolic link was created to another file system object.	link %s target %s
FILE_Pipe	***pipe()***	An unnamed pipe was created.	read descriptor %d write descriptor %d
FILE_Mknod	***mknod()***	A file system object was created (special file, FIFO, and so on).	mode %o device %d filename %s
FILE_Dupfd	***fcntl()***	A file descriptor is duplicated.	original file descriptor %d new file descriptor %d
FS_Extend	***fscntl()***	A file system is extended.	virtual file system %d command %d
FS_Mount	***mount()***	A file system was mounted.	object %s stub %s
FS_Umount	***umount()***	A file system was unmounted.	object %s stub %s

Table G.5 cont. File System Events

Audit Event	System Call	Description	Detailed Record Info
FILE_Acl	***chacl()***	A file system objects ACL was changed.	filename %s ACL %A
FILE_Facl	***fchacl()***	A file descriptors ACL was changed.	file descriptor %d ACL %A
FILE_Chpriv	***chpriv()***	A files privilege control list was changed.	filename %s privilege control list %P
FILE_Fchpriv	***fchpriv()***	A file descriptors privilege control list was changed.	file descriptor %d privilege control list %P
FS_Chdir	***chdir()***	The current working directory was changed.	directory to change to %s
FS_Fchdir	***fchdir()***	The current working directory was changed using a file descriptor.	file descriptor %d
FS_Chroot	***chroot()***	The root directory was changed.	directory to change to %s
FS_Rmdir	***rmdir()***	A directory was removed.	directory %s
FS_Mkdir	***mkdir()***	A directory was created.	directory %s mode %o
FILE_Utimes	***utimes()***	Change the modification and access times on a file.	filename %s

Table G.6: System V IPC Events

Audit Event	*System Call*	Description	Detailed Record Info
MSG_Create	***msgget()***	A new message queue is created.	key %d flag %o message id %d
MSG_Read	***msgrcv()***	A message is received from a message queue.	message queue id %d requester uid %d process id %d
MSG_Write	***msgsnd()***	A message is sent on a message queue.	message queue id %d
MSG_Delete	***msgctl()***	A message queue is removed.	message queue id %d
MSG_Owner	***msgctl()***	A message queues ownership or access rights were changed.	message queue id %d owner %d group %d mode %o
MSG_Mode	***msgctl()***	A message queues access rights were queried.	message queue id %d mode %o
SEM_Create	***semget()***	A new semaphore set was created.	key %d number of semaphores %d semaphore flag %o semaphore id %d
SEM_Op	***semop()***	One or more semaphores were incremented or decremented.	semaphore id %d

Table G.6 cont. System V IPC Events

Audit Event	System Call	Description	Detailed Record Info
SEM_Delete	***semctl()***	A semaphore set was deleted.	semaphore id %d
SEM_Owner	***semctl()***	The access rights or ownership for a semaphore set were changed.	semaphore id %d owner %d group %d mode %o
SEM_Mode	***semctl()***	A semaphore set's access rights were queried.	semaphore id %d mode %o
SHM_Create	***shmget()***	A new shared memory segment was created.	key %d size %d flags %o shared memory id %d
SHM_Open	***shmat()***	A shared memory segment was opened.	shared memory id %d
SHM_Close	***shmctl()***	A shared memory identifier was removed.	shared memory id %d
SHM_Owner	***shmctl()***	The ownership or access rights of the shared memory segment were changed.	shared memory id %d owner %d group %d mode %o
SHM_Mode	***shmctl()***	The access rights of a shared memory segment were queried.	shared memory id %d mode %o

Table G.7: TCP/IP Events

Audit Event	System Call	Description	Detailed Record Info
TCP_ksocket	***socket()***	Create a communications end-point.	fd %d domain %s type %s protocol %s
TCP_ksocketpair	***socketpair()***	Create a pair of connected sockets.	fd_1 %d fd_2 %d Domain %s Type %s Protocol %s
TCP_ksetopt	***setsockopt()***	Change or query a socket's attributes.	fd %d Protocol %s Option %d Value %d
TCP_kbind	***bind()***	Bind a socket to a port.	fd %d endpoint %S
TCP_klisten	***listen()***	Listen on a port.	fd %d limit %d
TCP_kconnect	***connect()***	A TCP connection has been created.	fd %d remote socket and user %L

Table G.7 cont. TCP/IP Events

Audit Event	System Call	Description	Detailed Record Info
TCP_kaccept		Accepts a connection from a client.	fd %d local %S remote %L
TCP_kshutdownt		Connection has been broken.	fd %d reason %s

Table G.8: Audit System Events

Audit Event	System Call	Description	Detailed Record Info
AUD_it	***audit()***	An audit subsystem request.	command %d argument %d
AUD_Bin_Def	***auditbin()***	Changes to bin mode auditing.	command %d current fd %d next fd %d bin size %d
AUD_Events	***auditevents()***	An audit events request.	command %d
AUD_Lost_Recs		A count of lost audit records.	records %d
AUD_Objects	***auditobj()***	An object auditing request.	command %d
AUD_Proc	***auditproc()***	A process auditing request.	pid %d command %d
PROC_Sysconfig	***sysconfig()***	A system configuration attribute has been modified.	request %x

Table G.9: Logical Volume Manager Events

Audit Event	Command / System Call	Description	Detailed Record Info
LVM_ChangeLV	***lchangelv***	A logical volume was changed.	logical volume id %s
LVM_ChangeVG	***lchangepv*** ***linstallpv***	A volume group was changed.	volume group id %s
LVM_CreateLV	***lcreatelv***	A logical volume was created.	logical volume id %s
LVM_CreateVG	***lcreatevg***	A volume group was created.	volume group id %s
LVM_DeleteVG	***ldeletepv***	A volume group was deleted.	volume group id %s
LVM_DeleteLV	***rmlv***	A logical volume was deleted.	logical volume id %s
LVM_VaryoffVG	***lvaryoffvg***	A volume group was varied off.	volume group id %s
LVM_VaryonVG	***lvaryonvg***	A volume group was varied on.	volume group id %s
LVM_AddLV	***hd_cfg()***	A logical volume was added to a volume group.	volume group id %x:%x logical volume index %d
LVM_KDeleteLV	***hd_cfg()***	The kernel deleted a logical volume.	volume group id %x:%x logical volume index %d
LVM_ExtendLV	***hd_cfg()***	A logical volume was extended in size.	volume group id %x:%x logical volume minor number %d
LVM_ReduceLV	***hd_cfg()***	A logical volume was reduced in size.	volume group id %x:%x logical volume minor number %d message %s

Table G.9 cont. Logical Volume Manager Events

Audit Event	Command / System Call	Description	Detailed Record Info
LVM_KChangeLV	***hd_cfg()***	A logical volume was changed.	volume group id %x:%x logical volume minor number %d message %s
LVM_AvoidLV	***hd_cfg()***	A physical volume is being avoided.	volume group id %x:%x logical volume minor number %d message %s
LVM_MissingPV	***hd_cfg()***	A physical volume is missing.	volume group id %x:%x physical volume index %d
LVM_AddPV	***hd_cfg()***	A physical volume was added to the system.	volume group id %x:%x physical volume device (major, minor) %d
LVM_AddMissPV	***hd_cfg()***	A missing physical volume was added.	volume group id %x:%x physical volume index %d
LVM_DeletePV	***hd_cfg()***	A physical volume was deleted.	volume group id %x:%x physical volume index %d
LVM_RemovePV	***hd_cfg()***	A physical volume was removed.	volume group id %x:%x physical volume index %d
LVM_AddVGSA	***hd_cfg()***	A vgsa area for a physical volume.	volume group id %x:%x physical volume index %d
LVM_DeleteVGSA	***hd_cfg()***	A vgsa area for a physical volume.	volume group id %x:%x physical volume index %d
LVM_SetupVG	***hd_cfg()***	A volume group was setup.	volume group id %x:%x
LVM_DefineVG	***hd_cfg()***	A volume group was defined.	volume group id %x:%x
LVM_ChgQuorum	***hd_cfg()***	Change the quorum count for a volume group.	volume group id %x:%x message %s
LVM_Chg1016	***hd_cfg()***	Change the number of partitions allowed on a disk.	volume group id %x:%x new factor value %d
LVM_UnlockDisk	***hd_cfg()***	Release the reserve on all the disks in the VG.	volume group id %x:%x
LVM_LockDisk	***hd_cfg()***	Regain the reserve on all the disks in the VG.	volume group id %x:%x

Table G.10: FSO Access Events

Audit Event	File	Description
S_ENVIRON_WRITE	***/etc/data.shared/environ***	A write occurred to the ***environ*** file.
S_GROUP_WRITE	***/etc/data.shared/etc.group***	A write occurred to the ***etc.group*** file.
S_LIMITS_WRITE	***/etc/data.shared/limits***	A write occurred to the ***limits*** file.
S_LOGIN_WRITE	***/etc/security/login.cfg***	A write occurred to the **login.cfg** file.
S_PASSWD_READ	***/etc/data.shared/passwd***	A read occurred from the ***passwd*** file.
S_PASSWD_WRITE	***/etc/data.shared/passwd***	A write occurred to the ***passwd*** file.
S_USER_WRITE	***/etc/data.shared/user***	A write occurred to the ***user*** file.
AUD_CONFIG_WR	***/etc/security/audit/config***	A write occurred to the ***audit/config*** file.

Table G.11: User And Administrative Commands

Audit Event	Command	Description	Detailed Record Info
SRC_Addserver	***addserver***	A new subserver has been added to SRC.	subserver attributes %s
SRC_Addssys	***addssys***	An SRC subsystem has been added to the ODM.	subsystem attributes %s
AT_JobAdd	***at***	An at job was submitted to the system.	filename %s user %s time%s
AT_JobRemove	***at*** ***cron***	An at job was removed from the system.	filename %s user %s
BACKUP_Export	***backbyinode*** ***rdump***	A backup of a file system by inode is being performed.	status %s
DEV_Configure	***cfgmgr*** ***mkdev***	A device has been configured.	device or error %s
DEV_Change	***chdev*** ***mkdev***	A change was made to an existing device.	parameters %s
GROUP_Change	***chgroup*** ***chsec*** ***mkgroup*** ***mkuser*** ***chgrpmem***	The attributes of a group have been modified.	group %s message %s
PORT_Change	***chsec***	The attributes of a port were modified.	port %s command line %s
SRC_Chserver	***chserver***	An SRC subserver has been modified.	subserver attributes %s
SRC_Chssys	***chssys***	The attribute of an SRC subsystem have been modified.	subsystem attributes %s
USER_Change	***chuser*** ***chgroup*** ***chsec*** ***chfn*** ***chsh***	The attributes of a user have been modified.	user %s message %s
CRON_Start	***cron***	The cron daemon began processing for a particular time range.	action %s command or file %s time %s
CRON_Finish	***cron***	The cron daemon completed execution for a particular time range.	user %s process id %s time %s
CRON_JobRemove	***crontab***	A **CRON** job was removed from the system.	filename %s user %s time %s
CRON_JobAdd	***crontab***	A **CRON** job was added to the system.	filename %s user %s time %s
ENQUE_admin	***enq***	An administrative request has been made.	queue %s device %s request %s mail address %s action %s

Table G.11 cont. User And Administrative Commands

Audit Event	Command	Description	Detailed Record Info
TCPIP_connect	***ftp*** ***ftpd*** ***rexec*** ***rexecd*** ***setclock*** ***telnet*** ***telnetd***	TCP/IP connection established.	protocol %s from address %s service %s open/close %s message %s
TCPIP_data_out	***ftp*** ***ftpd***	TCP/IP data sent.	from address %s local file %s remote file %s message %s
TCPIP_data_in	***ftp*** ***ftpd***	TCP/IP data received.	from address %s local file %s remote file %s message %s
TCPIP_access	***ftp*** ***ftpd*** ***rexecd*** ***rshd***	An access query was made.	from address %s service %s user %s result %s
GROUP_User	***grpck***	A non-existent user was removed from a group.	username %s group name %s
GROUP_Adms	***grpck***	A non-existent admin user was removed from a group in ***/etc/security/group***.	admin username %s group name %s
INIT_Start	***init***	The init process has started a subprocess.	process id %d command %s
INIT_End	***init***	A subprocess of init has ended.	process id %d status %d
INSTALLP_Inst	***installp***	An LPP has been installed on the system.	option name %s level %s status %s
USER_Logout	***logout***	A user logged off from the system.	tty %s
DEV_Create	***mkdev***	A device was created.	parameters or error %s
DEV_Start	***mkdev***	A device was started.	parameters or error %s
GROUP_Create	***mkgroup***	A new group was created.	group %s message %s
USER_Create	***mkuser***	A user was created on the system.	username %s attributes %s
NVRAM_Config	***nvload***	Non-volatile RAM has been modified.	parameters or error %s
PASSWORD_Change	***passwd*** ***pwdadm*** ***su*** ***tsm***	A user changed her password.	username %s tty %s
PASSWORD_Flags	***pwdadm***	The *flags* attribute for a user's password has been modified.	username %s flags %s terminal %s

Table G.11 cont. User And Administrative Commands

Audit Event	Command	Description	Detailed Record Info
PASSWORD_Check	***pwdck***	An entry in the password file has been modified.	username %s attribute %s status %s
PASSWORD_Ckerr	***pwdck***	An unexpected error occurred while attempting to verify the password file.	username or file, error
ENQUE_exec	***qdaemon***	A queued job has been executed.	queue %s request %s host %s file %s mail address %s action %s
USER_Reboot	***reboot***	An administrator has rebooted the system.	username %s
USER_Exit	***rlogind*** ***telnetd***	The login shell has terminated.	message %s
DEV_Stop	***rmdev***	A device was stopped.	device or error %s
DEV_Unconfigure	***rmdev***	A device was unconfigured.	device or error %s
DEV_Remove	***rmdev***	A device was removed from the system.	device or error %s
GROUP_Remove	***rmgroup***	A group was removed.	group %s
SRC_Delserver	***rmserver***	An SRC subserver has been removed.	subserver type %s
SRC_Delssys	***rmssys***	An SRC subsystem has been removed.	subsystem %s
USER_Remove	***rmuser***	A user was removed from the system.	username %s
RESTORE_Import	***rrestore*** ***restbyinode***	A restore of a file system by inode is being performed.	status %s
SENDMAIL_Config	***sendmail***	An attempt was made to change the **SENDMAIL** configuration file.	config file %s
SENDMAIL_ToFile	***sendmail***	An attempt was made to send mail directly to a user file.	remote username %s filename %s
SENDMAIL_ToUser	***sendmail***	An attempt was made to send mail to a local user.	remote username %s local username %s
USER_SetGroups	***setgroups*** ***newgrp***	The effective group or groupset has been changed.	primary group %s group set %s
USER_SetEnv	***setsenv***	The privileged environment has been changed.	username %s new environment value %s
USER_Shell	***shell***	A new shell has been started.	tty %s
SRC_Start	***startsrc***	A subsystem was started by the System Resource Controller.	subsystem name %s
SRC_Stop	***stopsrc***	A subsystem terminal normally.	subsystem name %s
USER_SU	***su***	A user changes identities on the system.	new username %s
SYSCK_Check	***sysck***	A problem was found while verifying a file.	file %s message %s attribute %s
SYSCK_Update	***sysck***	An entry is being updated in the ***sysck.cfg*** database.	file %s attribute %s

Table G.11 cont. User And Administrative Commands

Audit Event	Command	Description	Detailed Record Info
SYSCK_Install	***sysck***	Post-installation verification has been performed for a file.	file %s lpp %s
SYSCK_Delete	***sysck***	A file is being uninstalled from the system.	file %s lpp %s
TCBCK_Check	***tcbck***	A problem was found while verifying a file.	file %s message %s attribute %s
TCBCK_Update	***tcbck***	An entry is being updated in the sysck.cfg database.	file %s attribute %s
TCBCK_Delete	***tcbck***	A file is being uninstalled from the system.	file %s lpp %s
USER_Login	***tsm***	A user logged on to the system.	user %s tty or failure message %s
PORT_Locked	***tsm***	This port was locked due to invalid login attempts.	port %s
USER_Check	***usrck***	A problem was found while verifying a user attribute.	username %s action %s status %s
USRCK_Error	***usrck***	An error was found in an entry in ***/etc/passwd***.	bad password entry %s message %s

Appendix H. Trusted Programs

Table H.1. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
accton	performs process-accounting procedures	Yes
acledit	edits the access control information of a file	No
aclget	displays the access control information of a file	No
aclput	sets the access control information of a file	No
aixlong	display the contents of a print queue in long format	No
aixshort	display the contents of a print queue in short format	No
aixterm	initializes an Enhanced X-Windows terminal emulator	No
aixv2long	display the contents of a print queue from an AIX Version 2 system in long format	No
aixv2short	display the contents of a print queue from an AIX Version 2 system in short format	No
aixwide	display the contents of a print queue in wide format	No
arp	displays and modifies address resolution	Yes
at	runs commands at a later time	Yes
atq	displays the queue of jobs waiting to be run	Yes
atrm	removes jobs spooled by the at command	No
audit	controls system auditing	Yes
auditbin	manages bins of audit information	Yes
auditcat	writes bins of audit records	Yes
auditmerge	combine multiple audit data files from one or more hosts into a single stream	Yes
auditpr	formats bin or stream audit records to a display device or printer	Yes
auditselect	selects audit records for analysis according to defined criteria	Yes
auditstream	creates a channel for reading audit records	Yes
automount	mounts NFS file systems automatically	No
awk	finds lines in files that match patterns and then performs specified actions on them	No
backbyinode	create a backup of a file system based on individual i-nodes	Yes
backbyname	create a backup of a file system based on individual file names	No
backup	backs up files and file systems	No
bellmail	sends messages to system users and displays messages from system users	Yes
bindprocessor	binds a process to a processor	No
bootinfo	determines and displays various boot information, including boot device type and boot device name	No
bsdlong	display the contents of a BSD Version 4.3 print queue in long format	No
bsdshort	display the contents of a BSD Version 4.3 print queue in short format	No
cancel	cancels requests to a line printer	No
cat	concatenates or displays files	No
cfginet	loads and configures an internet instance	No
cfglvdd	configure the Logical Volume Manager kernel process	No
cfg64	configuration method for the 64-bit process environment	No
cfgmgr	configures devices by running the programs specified in the Configuration Rules Object Class	Yes
cfgvg	configure a Volume Group for use	No
chcons	redirects the system console to a specified device or file to be effective on the next start of the system	Yes

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
chdev	changes the characteristics of a device	Yes
chfn	changes a user's gecos information	No
chfs	changes attributes of a file system	No
chgroup	changes attributes for groups	No
chgrpmem	changes the administrators or members of a group	No
chlang	changes the language settings for system or user	No
chlicense	changes the number of fixed licenses and the status of the floating licensing of the system	No
chlv	changes only the characteristics of a logical volume	No
chlvcopy	change the characteristics of a Logical Volume copy	No
chmod	changes file modes	No
chnfs	changes the configuration of the system to invoke a specified number of biod and nfsd daemons	No
chnfsexp	changes the options used to export a directory to NFS clients	No
chnfsmnt	changes the options used to mount a directory from an NFS server	No
chown	changes the owner or group associated with a file	No
chps	changes attributes of a paging space	No
chpv	changes the characteristics of a physical volume in a volume group	No
chque	changes the queue name	Yes
chqueuedev	changes the printer or plotter queue device names	Yes
chrctcp	modify /etc/rc.tcpip according to input from the user	No
chrole	changes role attributes	No
chsec	changes the attributes in the security stanza files	No
chsh	changes a user's login shell	Yes
chssys	changes a subsystem definition in the subsystem object class	No
chtcb	changes or queries the tcb attributes of a file	Yes
chtz	changes the TimeZoneInfo (TZ) environment variable in the */etc/environment* file	No
chvg	sets the characteristics of a volume group	No
chvirprt	changes the attribute values of a virtual printer	Yes
clvmd	HACMP Concurrent Logical Volume Manager daemon This feature is not part of the C2 evaluated system.	Yes
copyrawlv	copy a raw Logical Volume to a new location	No
cp	copies files	No
cplv	copies the contents of a logical volume to a new logical volume	No
crfs	adds a file system	No
cron	runs commands automatically	Yes
cronadm	lists or removes CRONTAB or AT jobs	Yes
crontab	submits, edits, lists, or removes CRON jobs	Yes
cut	writes out selected bytes, characters, or fields from each line of a file	No
digest	converts an input file of ASCII characters into a binary file	Yes
dspmsg	displays a selected message from a message catalog	No
deflvm	define Logical Volume Manager configuration data for CFGMGR	No
df	reports information about space on file systems	No
disable	disables printer queue devices	No
defragfs	increases a file system's contiguous free space	No

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
deftunnel	defines and configures a tunnel device	No
devinstall	installs software support for devices	Yes
diskusg	generates disk accounting data by user ID	Yes
ds_form	front-end to DS_RSLT, generates the list of available search indexes.	Yes
ds_reslt	processes the results of the ds_rslt and return them in HTML format.	Yes
dspdiagtask	online diagnostic monitor - see if specified diagnostic is running	No
dtconfig	enables or disables the desktop auto-start	No
dump_smutil	utilities to put output from SYSDUMPDEV in cmd_to_discover format	No
egrep	searches a file for a pattern	No
enable	enables printer queue devices	No
enq	enqueues a file	Yes
entstat	provides ethernet device statistics	Yes
entstat.phxent	provides device statistics particular to F/C 2968	Yes
errpt	lists and formats the stanzas in the error log	Yes
exec_shutdown	The real shutdown	Yes
exportvg	exports the definition of a volume group from a set of physical volumes	No
extendlv	increases the size of a logical volume by adding unallocated physical partitions from within the volume group	No
extendvg	adds physical volumes to a volume group	No
fgrep	searches a file for a literal string	No
format	formats either diskettes or read/write optical media disks	Yes
fsck	checks file system consistency and interactively repairs the file system	No
ftp	transfers files between a local and remote a host	Yes
ftpd	provides the server function for the Internet FTP protocol	Yes
fuser	identifies processes using a file or file structure	Yes
getlvcb	reads parameters from the logical volume control block	No
getlvname	generates or checks a logical volume name	Yes
getlvodm	get volume group and logical volume data from the Object Data Manager (ODM)	No
getvgname	generates a volume group name	Yes
grep	searches a file for a pattern	No
groups	displays group membership	No
grpck	verifies the correctness of a group definition	Yes
head	displays the first few lines or bytes of a file or files	No
host	resolves a host name into an Internet address or an Internet address into a host name	Yes
hostent	directly manipulates address-mapping entries in the system configuration database	No
hostname	sets or displays the name of the current host system	No
hostnew	alternative form of IP address to hostname mapping tool, i.e. different form of the 'host' command	Yes
httpd	The HTTP server daemon proper - receives and processes HTTP protocol.	Yes
id	displays the system identifications of a specified user	No
importvg	imports a new volume group definition from a set of physical volumes	No
inetd	provides Internet service management for a network	Yes
installp	installs available software products in a compatible installation pattern	No
ipcs	reports interprocess communication facility status	Yes
ipcrm	removes message queue, semaphore set, or shared memory identifiers	No

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
ipl_varyon	vary on the root volume group during first phase IPL	Yes
istat	examines i-nodes	No
killall	cancels all processes except the calling process	Yes
ksh	invokes the Korn shell	No
lchangelv	changes the attributes of a logical volume	Yes
lchangepv	changes the attributes of a physical volume in a volume group	Yes
lchangevg	changes the attributes of a volume group	Yes
lchlvcopy	change the online mirror backup status of a logical volume	Yes
lcreatelv	creates an empty logical volume in a specified volume group	Yes
lcreatevg	creates a new volume group and installs the first physical volume	Yes
ldeletelv	deletes a logical volume from its volume group	Yes
ldeletepv	deletes a physical volume from a volume group	Yes
lextendlv	extends a logical volume by a specified number of partitions	Yes
linstallpv	installs a physical volume into a volume group	Yes
lmigratepp	moves a physical partition to a specified physical volume	Yes
lmktemp	create a logical volume template file	No
ln	links files	No
locale	writes information about current locale or all public locales	No
logname	displays login name	No
logout	stops all processes on a port	Yes
lpd	provides the remote printer server on a network	Yes
lquerylv	returns information on the logical volume specified	Yes
lquerypv	returns information on the physical volume specified	Yes
lqueryvg	returns information on the volume group specified	Yes
lqueryvgs	queries the volume groups of the system and returns information for groups that are varied on-line	Yes
lreducelv	reduces a logical volume by a specified number of partitions	Yes
lresynclp	synchronizes all physical partition copies of a logical partition	Yes
lresynclv	synchronizes all physical partition copies of a logical partition	Yes
ls	displays the contents of a directory	No
ls_admin	displays and edits the license server database	No
lsadpnm	displays all available communications adapter names	No
lsallq	lists the names of all configured queues	No
lsallqdev	lists all configured printer and plotter queue device names within a specified queue	No
lsattr	displays attribute characteristics and possible values of attributes for devices in the system	No
lsaudit	display audit classes currently defined	Yes
lscfg	displays configuration, diagnostic, and vital product data information	Yes
lsconn	displays the connections a given device, or kind of device, can accept	No
lsdev	displays devices in the system and their characteristics	No
lsfs	displays the characteristics of file systems	No
lsgroup	displays the attributes of groups	No
lsitab	lists records in the */etc/inittab* file	No
lsjfs	displays the attributes of journaled file systems	No
lslpp	lists software products	No

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
lslv	displays information about a logical volume	Yes
lsmle	lists various cultural convention items from the ODM	No
lsnfsexp	displays the characteristics of directories that are exported with the Network File System	No
lsnfsmnt	displays the characteristics of NFS mountable file systems	No
lsparent	displays the possible parent devices that accept a specified connection type or device	No
lsps	displays the characteristics of paging spaces	No
lspv	displays information about a physical volume within a volume group	Yes
lsque	displays the queue stanza name	No
lsquedev	displays the device stanza name	No
lsresource	displays bus resources for available devices in the system and recommends attribute values for bus resource resolution	Yes
lsrole	displays role attributes	No
lssec	lists the attributes in the security stanza files	No
lssrc	gets the status of a subsystem, a group of subsystems, or a subserver	Yes
lsuser	displays attributes of user accounts	No
lsvfs	lists entries in the */etc/vfs* file	No
lsvg	displays information about volume groups	Yes
lsvgfs	list the file systems contained in a volume group	Yes
lsvirprt	displays the attribute values of a virtual printer	No
lvaryoffvg	vary off a volume group	Yes
lvaryonvg	vary on a volume group	Yes
lvchkmajor	check and return the major number for a volume group	No
lvgenmajor	return a major number for a volume group	Yes
lvgenminor	return a minor number for a volume group or logical volume	Yes
lvlstmajor	list the range of available major numbers	No
lvmmsg	output a logical volume command message	No
lvrelmajor	release a major number for a volume group	Yes
lvrelminor	release a minor number for a logical volume	Yes
mergedev	copy devices from first phase IPL ramdisk to root directory	No
mesg	permits or refuses write messages	Yes
migfix	calculate moves needed to migrate a physical partition	No
migratepv	moves allocated physical partitions from one physical volume to one or more other physical volumes	No
mirrorvg	mirrors all the logical volumes that exist on a given volume group	No
mkdev	adds a device to the system	Yes
mkgroup	creates a new group	No
mkitab	makes records in the */etc/inittab* file	No
mklv	creates a logical volume	No
mklvcopy	provides copies of data with the logical volume	No
mknfsexp	exports a directory to NFS clients	No
mknfsmnt	mounts a directory from an NFS server	No
mknod	creates a special file	Yes
mkps	adds an additional paging space to the system	No
mkque	adds a printer queue to a system	Yes

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
mkquedev	adds a printer queue device to a system	Yes
mkrole	creates new roles	No
mksysb	creates an installable image of the root volume group	No
mktcpip	sets the required values for starting TCP/IP on a host	No
mkuser	creates a new user account	No
mkvg	creates a volume group	No
mkvirprt	makes a virtual printer	Yes
mount	makes a file system available for use	Yes
namerslv	directly manipulates domain name server entries for local resolver routines in the system configuration database	No
newgrp	changes a user's real group identification	Yes
netstat	shows network status	Yes
nfsstat	displays statistical information about the Network File System (NFS) and Remote Procedure Call (RPC) calls	Yes
od	displays files in a specified format	No
odmget	retrieves objects from the specified object classes into an odmadd input file	No
oslevel	reports the latest installed maintenance level of the system	No
pac	prepares printer/plotter accounting records	Yes
passwd	changes a user's password	Yes
penable	enables or reports the availability of login ports	Yes
ping	sends an echo request to a network host	Yes
pioattred	provides a way to format and edit attributes in a virtual printer	Yes
piobe	printer job manager for the printer backend	Yes
piochdfq	changes the default print queue	No
piodmgrsu	run PIODMGR as the root user	Yes
pioevattr	shows virtual printer attributes	No
piofontin	copies fonts from a multilingual font diskette	Yes
piomkapqd	builds a SMIT dialog to create print queues and printers	Yes
piomkpq	creates a print queue	Yes
piomisc_base	printer queue and job query/maintenance routines	No
pioout	printer backend's device driver interface program	Yes
piopredef	creates a predefined printer data-stream definition	Yes
plotbe	plots HP-GL files to a plotter device	No
ps	shows current status of processes	Yes
putlvcb	writes logical volume control block parameters into the logical volume control block	Yes
putlvodm	put volume group and logical volume data into the Object Data Manager	Yes
pwdadm	administers users' passwords	No
pwdck	verifies the correctness of local authentication information	Yes
qadm	performs system administration functions for the printer spooling system	No
qchk	displays the status of a print queue	No
qdaemon	schedules jobs enqueued by the ENQ command	Yes
qpri	prioritizes a job in the print queue	No
qstatus	provides printer status for the print spooling system	No
quotacheck	checks file system quota consistency	No
rcp	transfers files between a local and a remote host or between two remote hosts	Yes

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
rc.tcpip	initializes network daemons at each system restart	No
reboot	restarts the system	Yes
readlvcopy	Read/copy all or a portion of a logical volume	No
redefinevg	redefines the set of physical volumes of the given volume group in the device configuration database	No
reducevg	removes physical volumes from a volume group	No
refresh	requests a refresh of a subsystem or group of subsystems	Yes
reorgvg	reorganizes the physical partition allocation for a volume group	No
restbyinode	restore one or more files from a backup which was performed by inode	Yes
restbyname	Restore by name	No
restore	copies previously backed-up file systems or files, created by the backup command, from a local device	No
restvg	restores the user volume group and all its containers and files, as specified in the */tmp/vgdata/vgname/vgname.data* file contained within the backup image created by the SAVEVG command	No
rexec	executes commands one at a time on a remote host	Yes
rexecd	provides the server function for the REXEC command	Yes
rlogin	connects a local host with a remote host	Yes
rlogind	provides the server function for the rlogind daemon	Yes
rm	removes (unlinks) files or directories	No
rmdev	removes a device from the system	Yes
rmfs	removes a file system	No
rmgroup	removes a group	No
rmitab	removes records in the */etc/inittab* file	No
rmlv	removes logical volumes from a volume group	No
rmlvcopy	removes copies from a logical volume	No
rm_mlcache_file	remove the OSLEVEL command cache file	Yes
rmnfsexp	unexports a directory from NFS clients	No
rmnfsmnt	removes an NFS mount	No
rmque	removes a printer queue from the system	Yes
rmquedev	removes a printer or plotter queue device from the system	Yes
rmrole	removes a role	No
rmuser	removes a user account	No
rmvirprt	removes a virtual printer	Yes
route	manually manipulates the routing tables	Yes
rsh	executes the specified command at the remote host or logs into the remote host	Yes
rshd	provides the server function for remote command execution	Yes
ruser	directly manipulates entries in three separate system databases that control foreign host access to programs	No
savebase	saves information about base-customized devices in the Device Configuration database onto the boot device	No
savevg	finds and backs up all file belonging to a specified volume group	No
securetcpip	enables the operating system network security feature	No
sed	provides a stream editor	No

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
semutil	create or remove the **SENDMAIL** queue directory semaphore	Yes
sendmail	routes mail for local or network delivery	Yes
setclock	sets the time and date for a host on a network	Yes
setgroups	resets a session's process group set	Yes
setmaps	sets terminal maps or code set maps	No
setsenv	resets the protected state environment of a user	Yes
shell	executes a shell with the user's default credentials and environment	Yes
showled	change the values on the front panel LED display	No
shutdown	ends system operation	No
sm_inst	displays packages and filesets available on install media	No
smitacl	Reads roles database to determine if user is permitted to view designated SMIT panels	Yes
sort	sorts files, merges files that are already sorted, and checks files to determine if they have been sorted	No
splitlvcopy	splits copies from one logical volume and creates a new logical volume from them	No
splp	changes or displays printer driver settings	Yes
srcmstr	starts the system resource controller	Yes
startsrc	starts a subsystem, a group of subsystems, or a subserver	Yes
stopsrc	stops a subsystem, a group of subsystems, or a subserver	Yes
strings	finds the printable strings in an object or binary file	No
su	changes the user ID associated with a session	Yes
swapon	specifies additional devices for paging and swapping	Yes
swcons	redirects, temporarily, the system console output to a specified device or file	Yes
synclvodm	synchronizes or rebuilds the logical volume control block, the device configuration database, and the volume group descriptor areas on the physical volumes	No
syncvg	synchronizes logical volume copies that are not current	No
sysck	checks the inventory information during installation and update procedures	Yes
sysdumpdev	changes the primary or secondary dump device designation in a running system	No
sysdumpstart	provides a command line interface to start a kernel dump to the primary or secondary dump device	No
tail	writes a file to standard output, beginning at a specified point	No
tcbck	audits the security state of the system	Yes
tcpdump	prints out packet headers	No
tee	displays the output of a program and copies it into a file	No
termdef	queries terminal characteristics	No
tctl	gives subcommands to a streaming tape device	No
telnet	invokes the telnet server daemon	Yes
telnetd	provides the server function for the TELNET protocol	Yes
timed	invokes the time server daemon	No
timedc	returns information about the timed daemon	Yes
tokstat	shows token-ring device driver and device statistics	Yes
tokstat.cstok	device specific support for tokstat	Yes
tr	translates characters	No
traceroute	prints the route that IP packets take to a network host	Yes
tracesoff	turns off tracing of a subsystem, a group of subsystems, or a subserver	No
traceson	turns on tracing of a subsystem, a group of subsystems, or a subserver	No

Table H.1. cont. Trusted Programs

User Commands		
Name	**Brief Description**	**Setuid0**
tsm	provides terminal state management	Yes
tstresp	test for a valid yes or no response in the current locale	No
tvi	provides a trusted editor with a full screen display	No
umount	unmounts a previously mounted file system, directory, or file	Yes
unmirrorvg	removes the mirrors that exist on volume groups or specified disks	No
updatelv	update Database for Logical Volume in the Volume Group	No
updatevg	update Database for Volume Group	No
users	displays a compact list of the users currently on the system	No
usrck	verifies the correctness of a user definition	Yes
varyoffvg	deactivates a volume group	No
varyonvg	activates a volume group	Yes
w	prints a summary of current system activity	Yes
wall	writes a message to all users that are logged in	No
watch	observes a program that may be untrustworthy	Yes
wc	counts the number of lines, words, and bytes in a file	No
who	identifies the users currently logged in	No
whoami	displays your login name	No
wsmappletcfg	configuration utility for wsm	No
X	starts the X server.	No
xinit	initializes the X Window System.	No
xlock	locks the local X display until a password is entered	Yes
xpowerm	starts Power Management GUI (Graphical User Interface) Utility	No
xterm	provides a terminal emulator for the X Window System	No

APPENDIX I: EVALUATOR COMMENTS

Testing Documentation

The team found the test documentation and supporting design documentation to be useful, well written and organized. The Test Matrix Document (TMD) followed the Architecture Summary Document and the Interface Summary Document closely. The TMD also was written in HTML and provided links from high level matrices to lower level matrices. This aided understanding and quickened test coverage analysis.

Configuration Management

IBM's use of the Configuration Management Version Control (CMVC) system was used to coordinate and track software versions and associated documents and manuals. Any and all modifications were recorded, tracked, and resolved as defects against the code, user manuals, administrator manuals, test documents, and test code.